
# DEVELOPING RANDOMIZED ENHANCED ACTIVE VM LOAD BALANCING ALGORITHM FOR CLOUD COMPUTING

**ASTER ALEMU**

**MASTER OF SCIENCE**

**ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY**

**DECEMBER 2018**


# DEVELOPING RANDOMIZED ENHANCED ACTIVE VM LOAD BALANCING ALGORITHM FOR CLOUD COMUTING

**By**

**ASTER ALEMU**

A Thesis Submitted to

The Department of software engineering for the Partial Fulfillment of the Requirements for the Degree of Master of Science in Software engineering

**ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY**

**DECEMBER 2018**

# Declaration

I, the undersigned, declare that this thesis is my original work and has not been presented for a degree in any other university, and that all source of materials used for the thesis have been duly acknowledged.

**Declared by:**

Name: _____

Signature: _____

Date: _____

**Confirmed by advisor:**

Name: _____

Signature: _____

Date: _____

# Certificate

This is to certify that the thesis prepared by Aster Alemu entitled "Developing Randomized Enhanced Active VM Load Balancing Algorithm for Cloud Computing" and submitted in fulfillment of the requirements for the Degree of Master of Science complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Board:

| | |
|---|---|
| Examiner: | Signature, Date: |
| Examiner: | Signature, Date: |
| Thesis Advisor: | Signature, Date: |

# Abstract

Computing is being transformed into a model in which users access services based on their requirements without regarding where the services are hosted. A number of computing models have promised to deliver the computing services and cloud computing is one of them. Which is an emerging computing paradigm which promised to provide opportunities for delivering a variety of computer applications in a way that has not been experienced before. A number of users switching to cloud services is increasing day by day, which in turn requires better performance of cloud. There are many issues that needs to be addressed for better performance of cloud computing. One of the major challenges of cloud computing is load balancing. Distribution of work load evenly across the virtual machine is called load balancing.

A technique used to provide load balancing for cloud computing is called load balancing algorithm and many researchers proposed different load balancing algorithms to improve the performance of cloud computing but this existing algorithms have their own deficiency as a result of this there is a need to have improved load balancing algorithm for cloud computing. In this research we proposed enhanced load balancing algorithm by taking advantage of random algorithm and enhanced active monitoring load balancing algorithm in order to improve response time and resource utilization problem in load balancing for cloud computing. The algorithm considers the current load of each virtual machine and assign requests to the least load and not recently loaded virtual machine by avoiding scanning of the virtual machine list table again and again from its first index to find least loaded and not last used virtual machine.

In order to evaluate the performance of load balancing algorithm, we used cloudAanalyst simulator and for evaluation, response time and resource utilization parameter is considered.

Finally simulation result of proposed algorithm is compared with three well-known load balancing algorithms which are Round robin, Active monitoring and Enhanced active monitoring. From the experiment result, the proposed algorithm had better response time, and improved resource utilization than the other three algorithms.

**Keywords:** Load Balancing, Cloud Computing, Virtual Machine, Virtualization, Cloud Analyst

## Acknowledgments

First of all I thank God for guiding me and taking care of me all the time. I would like to express my profound sense of gratitude towards my advisor Dr.Dereje Yohannes for his able guidance, support and encouragement throughout the period this work was carried out.

I wish to express my considerable gratitude to my dear friend Betelehem Akmel and Eyob Samuel who gave their efforts and guidance at appropriate times without which it would have been very difficult on my part to finish this thesis.

# Table of Contents

# List of Tables

# List of Figures

## Acronyms and Abbreviations

| | |
|---|---|
| AMLB | Active Monitoring Load Balancing |
| DC | Data Center |
| DCC | Data Center Controller |
| EAMLB | Enhanced Active Monitoring Load Balancing |
| IAAS | Infrastructure as a Service |
| IDE | Integrated Development Environment |
| NIST | National Institute of Standards and Technology |
| PAAS | Platform as a Service |
| REAVM | Randomized Enhanced Active VM |
| RR | Round Robin |
| SAAS | Software as a Service |
| SLA | Service-Level Agreement |
| UB | User Base |
| VM | Virtual Machine |
| VMM | Virtual Machine Monitor |

# Chapter 1: Introduction

## 1.1 Overview of Cloud Computing

Computing is being transformed into a model which allow computing service users to access their computing services without regarding where those services are accommodated. A number of computing paradigms are available which intended to deliver computing services such as green computing, distributed computing, utility computing and cloud computing. From the listed computing models, cloud computing is relatively recent term even if it was developed upon some of the existing concepts and it is defined in many ways by different researchers and research organizations but the most and the common one is a definition given by a National Institute of Standards and Technology (NIST). So, based on NIST cloud computing refers to distribution of pool of resources among users through internet according to use. In cloud computing model, the user access service in pay per use base means the user only pay for the service they use [1]. There are three service delivery models in which cloud service providers deliver their computing service to the customer. Those are: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). To build cloud computing environment four deployment models are available: private Cloud, Public Cloud, Community Cloud and Hybrid Cloud[2, 4,7]. One of the major challenges of cloud computing is load balancing. Handling and servicing millions of requests from the users arriving at the data center controller and distributing the load evenly across all the available machines are called load balancing. Load balancing is aimed to help in having high resource utilization, to maximize accessibility of computing services, to maximize customer satisfaction, to minimize execution time and waiting time of task coming from different location to improve the performance, maintain system stability, build fault tolerance system and accommodate future modification. And one of the way to achieve this is by providing effective and efficient load balancing techniques, these techniques are called load balancing algorithms [3, 9].

There are two types of load balancing algorithms: static load balancing algorithm and dynamic load balancing algorithm. In static algorithms, load is assigned to different machines in view of virtual machines processing capabilities and this type of load balancing algorithms are mainly appropriate for homogeneous and stable environments. But, they are typically not

flexible and cannot match the dynamic changes to the attributes during the execution time [1,6]. In dynamic algorithms, load is assigned to different machines in view of the changes that happen in runtime conditions of the virtual machine and it is good in a heterogeneous and dynamic environments. However, the distribution attributes become more complex and dynamic, some of dynamic algorithms could become inefficient and cause more overhead, which in turn results the degradation of services performance [5]. Because of this, there is need to develop enhanced load balancing algorithm. In this research work we proposed developing randomized enhanced active VM load balancing algorithm for cloud computing by taking advantages of two existing load balancing algorithms which are: random and enhanced active monitoring algorithm. Again this algorithm checks the virtual machine is not recently used or not before assigning requests to avoid assigning one least loaded VM in a continuous manner and this helps to improve overall performance of the cloud by providing better resource utilization and minimum response time.

## 1.2 Statement of the Problem

The aim of cloud computing is to decrease capital and operational cost, improved performance in terms of response time ,data processing time, and better utilization of resource. One of the ways to do this is by providing efficient and effective VM load balancing techniques. Load balancing is distributing the load fairly across all the available machines and the main aim of load balancing is to help in achieving high resource utilization and user satisfaction and main issues of load balancing are throughput, resource utilization, response time and overhead associated [8].

There are various VM load balancing algorithms which are well-known on current cloud environment. The most popular algorithms are Round-robin which uses the round robin order for allocating job. It assign the first node randomly for the first request and all other jobs are assigned in a circular order. In this load balancing algorithm the incoming job should wait in the queue if there is no available VM and which in turn increase the response time [12]. In the case of Active Monitoring Load Balancing algorithm, information about the VM is maintained in an index table for VMs together with their current number of load. When a request arrives to data center then the data center controller question the load balancer for appropriate VM then the load balancer send id of least load VM to the data center (DC). In case if two VMs

have same number of allocation then the first VM is assigned and after that current count of that VM will be increased by one. When processing the task is finished then the current count of the node is decreased by one. But searching least loaded VM start from the first index each and every time, this result in additional overhead and which in turn maximize the response time. Also there is a situation in which one VM will be allocated in continuous manner if it is least loaded and this minimize resource utilization. Random algorithm is used to assign the selected job randomly to the available VM. This algorithm doesn't consider the status of the VM (it will not check the VM is under heavy load or low load) and this will result in maximum response time [11, 13]. So there is a huge gap in providing effective and efficient load balancing for cloud computing.

Therefore there is a need to have improved load balancing algorithm for cloud computing. So this thesis is amid to solve response time problem and resource utilization in cloud computing in order to improve the performance of cloud computing that enable cloud service provider to satisfy their service user and the service user can access their request within a short period of time.

## 1. 3 Objectives

### 1.3.1 General Objectives

The general objective of this research is to design a Randomized Active VM load balancing algorithm with enhanced features for cloud computing platforms.

### 1. 3.2 Specific Objectives

- Design load balancing algorithm for cloud computing.
- Plan and implement proposed algorithm.
- Test and evaluate the proposed algorithm using response time and resource utilization metrics
- Compare result of proposed algorithm with round robin, active monitoring and enhanced active monitoring algorithm.

## 1.4 Scope and Limitations

The main focus of this research is examine existing well-known load balancing algorithms and find their drawback, design and implement load balancing algorithm in order to solve their problem. For evaluation purpose we used three popular algorithms called round robin, active monitoring and enhanced active monitoring load balancing algorithm. Performance of the proposed algorithm is measured using cloudAnalyst simulator but not real experiments and it is proposed only to manage the incoming request arrived on the data center and assign the requests based on the current load of VMs. Above and beyond, this research work does not address other issues of cloud computing other than load balancing and also the load balancing algorithm will be developed under data center level that means the request type, size and routing between client node and data centers are not considered.

## 1.5 Significance of the Study

This research aims to address problem of some of the existing well known load balancing algorithm since the current load balancing algorithm have their own limitation for example round robin load balancing algorithm make requests to wait in the queue if there is no available VM. On the other hand active monitoring load balancing algorithm assign task in a continuous manner to the least loaded virtual machine and this cause unfair allocation tasks to the node [1]. On the other hand, enhanced active monitoring scan the index table starting from its first index to find least and not last used VM each and every time. Again this increase the associated overhead and which in turn increase response time. From this point of view the new algorithm is significant to solve such a problem in the existing load balancing algorithms.

## 1.6 Beneficiaries

The result of this research work help different organization which provide various cloud service to the end user in efficient manner. And the proposed load balancing algorithm is good to provide the cloud service with minimum response time and also it helps the cloud providers to utilize their resource efficiently. In addition to this the study will become a motivation for conducting other researches on this area.

## 1.7 Organization of the Document

The rest of this thesis is organized into five major chapters: The second chapter is literatures review around cloud computing in detail, existing load balancing algorithms contribution with their drawback and the third chapter discuss about the methodology which are used to conduct our research. In fourth chapter we discuss about the proposed algorithm including its flow chart and pseudo code. The fifth chapter is all about experimentation and results, therefore in this chapter the study tried to show the analysis of the experimentation in line with the discussion of the results obtained from the CloudAnalyst simulation tool. The last chapter is about conclusion and future works or recommendation.

# Chapter 2: Literature Review

This Chapter deals with review of important concepts concerning the research work. It provides detailed definition of cloud computing, its main characteristics, cloud computing service model and cloud computing deployment models. In addition to this it deals about cloud computing issues more specifically about load balancing issues in cloud computing with the existing load balancing algorithms and we also discuss about cloud simulators.

## 2.1 Overview of Cloud Computing

Computing is a process of using or utilizing different computer technology's to complete a given task and it enables computing service user to have access to different computing services without regarding where they are hosted. There are a number of computing models like grid computing, distributed computing, cluster computing and cloud computing. Among this computing models, cloud computing is relatively recent term even though it was built upon some of the existing concepts to provide opportunities for delivering a variety of computer applications in a way that has not been experienced before. Many researchers define cloud computing in a number of ways and the most and the common one is a definition given by National Institute of Standard and Technology (NIST). According to NIST Cloud computing is defined as a computing model which is used everywhere and provides convenient, on-demand access to a shared pool of computing resources such as networks, servers, storage, applications, etc. These resources can be dynamically allocated and released with minimum management effort or cloud computing service user can access their computing service without having interaction with the cloud service providers [15, 16].

The difference that cloud computing brings compared to the traditional computing models like grid computing, distributed computing, utility computing, or autonomic computing is to widen prospects across organizational boundaries in which cloud computing employs distributed resources to achieve application-level objectives by employing virtualization technologies at multiple levels to realize resource sharing and dynamic resource provisioning. Again cloud computing adopts a utility-based pricing or it provides opportunity in which users can only pay the service they use. In addition to this, service providers can maximize resource utilization and minimize their operating costs with on-demand resource provisioning and utility based pricing and this enables to experience direct cost benefits and it has the capability

to change a data center from a capital-intensive set up to a variable priced environment. The name "cloud computing" is used for this computing model because the information to be accessed is found in the cloud and cloud service user can access their service without having the limitation of being in a specific place [17, 40].

### 2.1.1 Cloud Computing Architecture

In general the architecture of a cloud computing environment can be divided into 4 layers: hardware layer, infrastructure layer, platform layer and application layer.

a. **Hardware layer**: it is responsible to manage the clouds physical resources like servers, switches, routers and others. This layer is implemented in data center and it contains a large number of servers which are organized and interconnected by using switches or routers [21, 40].
b. **Infrastructure layer:** this layer is also called virtualization layer, it is used to create a pool of storage, computing resources and uses virtualization technologies for dividing the physical resources by using Xen, KVM and VMware. This layer is a very important element of cloud computing, because dynamic resource assignment and many other key features are available only through virtualization technologies [21, 40].
c. **Platform layer:** which is built on top of virtualization layer and it consists of application frameworks and operating systems. The main responsibility of this layer is to minimize the load of deploying applications directly into VM containers [21, 40].
d. **Application layer:** this layer consists of the real cloud applications and it can leverage the automatic scaling feature to attain improved performance, accessibility and minimum operating cost as compared to traditional applications, such as dedicated server farms, the architecture of cloud computing is more modular[21, 40].

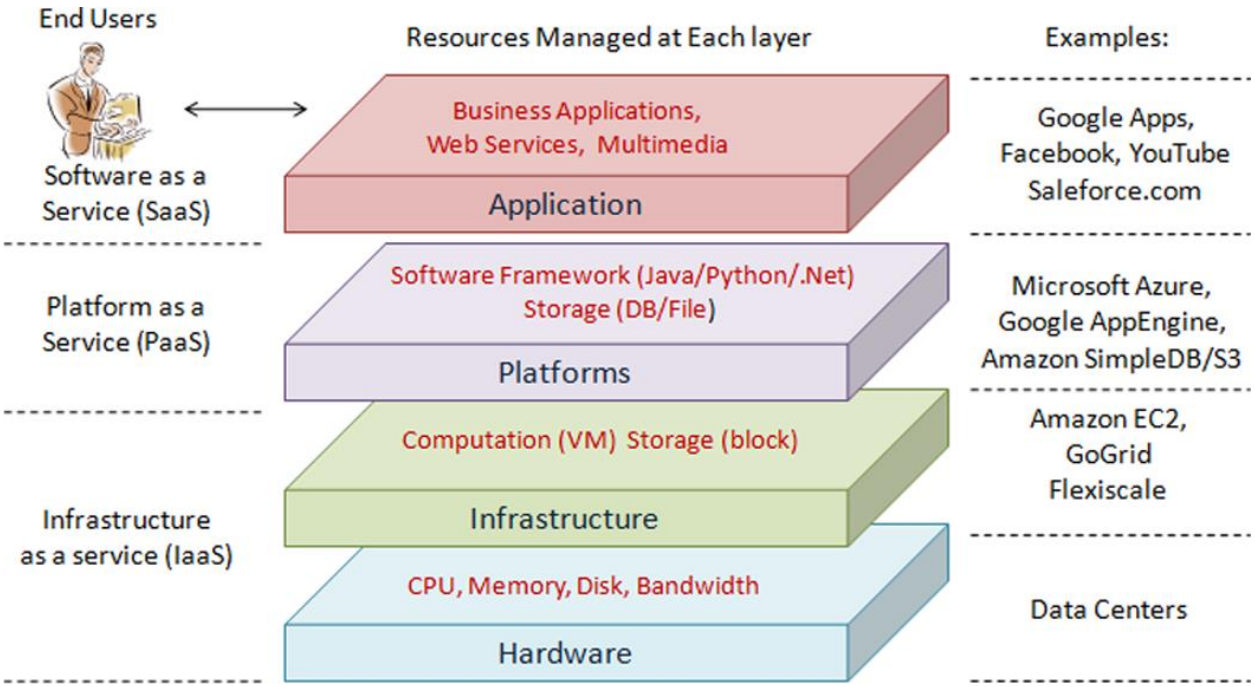



Figure 2. 1 Cloud computing architecture [21].

### 2.1.2 Characteristics of Cloud Computing

Advancement of cloud computing is vast with respect to personal uses and business uses. Users of cloud computing can utilize or maintain the online resources. Among several advantages or benefits few of them are discussed below [17].

**a. On-demand self-service**: cloud service users can access their own computing service without the help of other person or without the need to interact with the cloud service providers [17, 22].

**b. Broad network access**: the cloud services are available over the network and can be accessed by using standard mechanisms like heterogeneous thin or thick client platforms (e.g. Mobile phones, laptops, and PDAs) [17, 22].

**c. Rapid elasticity**: the cloud service provider can scale up or scale down their service based on the need of their customer without the quality of services being affected. [17, 22].

**d. Measured service**: Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g. storage, processing, bandwidth, and active user accounts). Resource usage can be

managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service [17, 22].

**e. Resource pooling**: cloud service provider can provide their computing service to multiple number of consumers at a time by using a multi-tenant model and there is no a sense of geographical or location dependency in which the user generally has no knowledge of the exact location of the provided services but may be able to specify location at a higher level of abstraction (e.g. country, state, or data center) [17, 22].

### 2.1.3 Cloud Service Model

In cloud computing environment, services are delivered in three different way. Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS)

**a. Software as a service (SaaS):** in this service delivery model, all applications are provided to the consumer and the consumers are not responsible for the underling infrastructure. Users can access service by using interfaces like web browser. Some of the examples that provide SaaS are Google, Salesforce, Microsoft, Zoho [18, 24].

**b. Platform as a service (PaaS):** in this service delivery model the customer can gain access of services that enable them to build applications and they are not responsible to download and install the software. They deploy all the application onto the cloud infrastructure. Also the consumer does not control network, servers, operating systems, or storage. Consumer controls all applications which they deploy. Google"s App Engine, Force.com, etc are some of the popular PaaS examples [18, 24].

**c. Infrastructure as a service (IaaS):** In this service consumer does not manage or control the underlying cloud infrastructure. In this service delivery model consumer are responsible to control operating systems, storage, and all applications which they deployed. Service providers control storing and processing capacity. Some common examples are Amazon, GoGrid, and 3 Tera [18].

### 2.1.4 Cloud Deployment Model

In order to build cloud computing environment to provide services to different user, four deployment models are available and each of them are discussed below.

**a. Private cloud:** is used to build the cloud infrastructure for an organization which requires to be operated exclusively. The deployments made inside the organization firewall's to keep their sensetive information [16, 19, 25].

**b. Public cloud:** in this model, the cloud infrastructure is available to the general public and the user data is publically visible. Public cloud vendors typically provide an access control mechanism for their users. [16, 19, 25].

**c. Community cloud:** in this model the cloud environment is built for several organizations which have a common goal and those organization share the services and it may be handled by the organizations or other party to smoothly access the services of cloud [16, 19, 25].

**d. Hybrid cloud:** it is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability. In this model client typically outsource non business critical information & processing to public cloud, while having control on critical information and data [16, 19, 25].

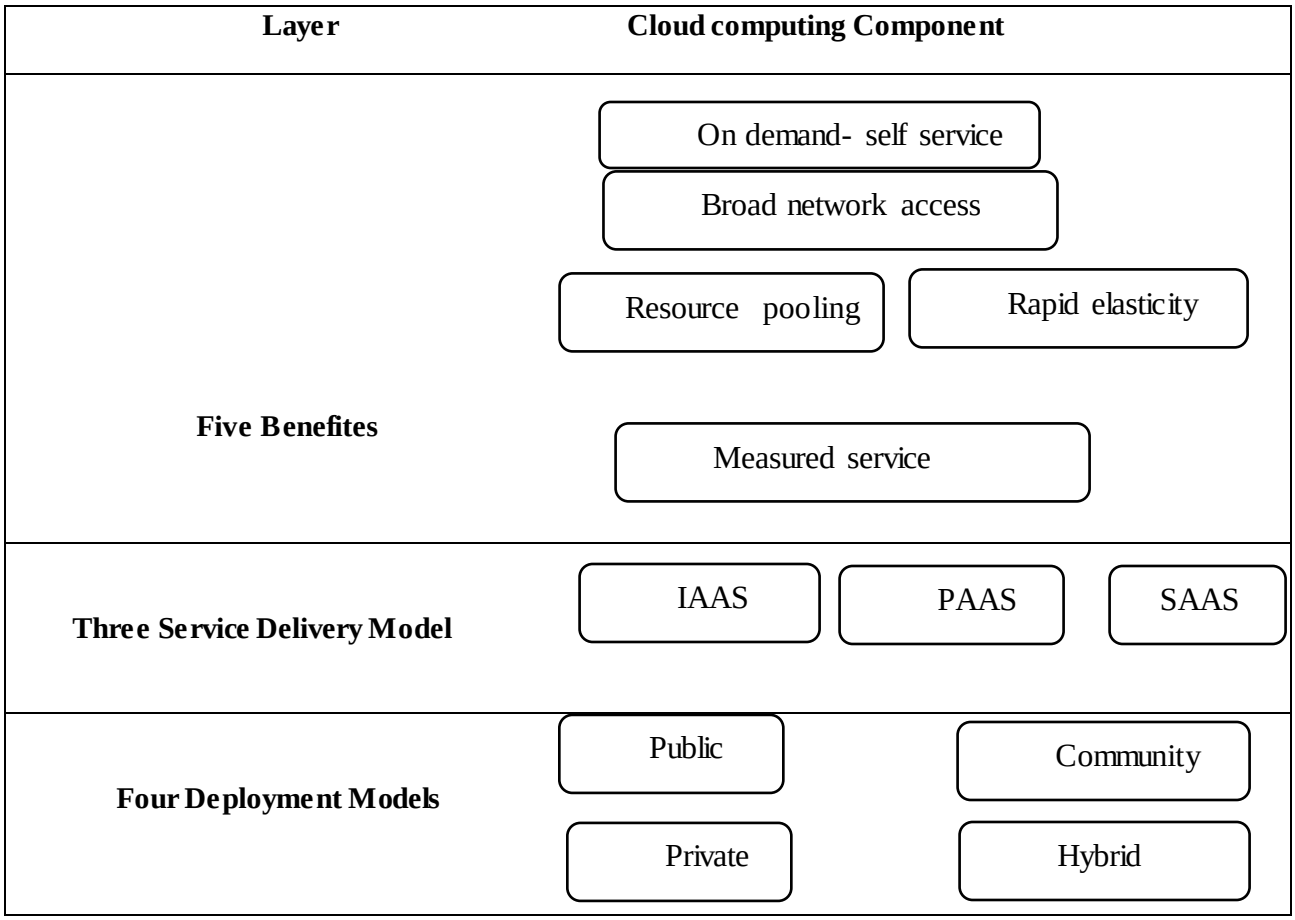



Figure 2. 2 Model of cloud computing

## 2.2 Virtualization

Virtualization is something which is not visible but it performs every activity that can be performed by real environment. It is the software operation of a hardware or a real machine which will provide various application as a real machine. By using different virtualization technology, cloud service user can access their computing services on the cloud.

Currently virtualization becomes the main foundation for cloud computing in order to provide a very accessible and virtualized resources. By abstracting and isolating the primary hardware, and networking resources in a solitary hosting environment, It enables to manage resource easily in cloud computing environment [5].

Resource sharing, rapid provisioning, and workload isolation can be performed in virtualization environment. A software called which directly runs on hardware and it is defined as the virtual machine manager to allow a number of operating systems to run on a

system at a time by providing necessar resources to every OS without a need to interact with each other[21].

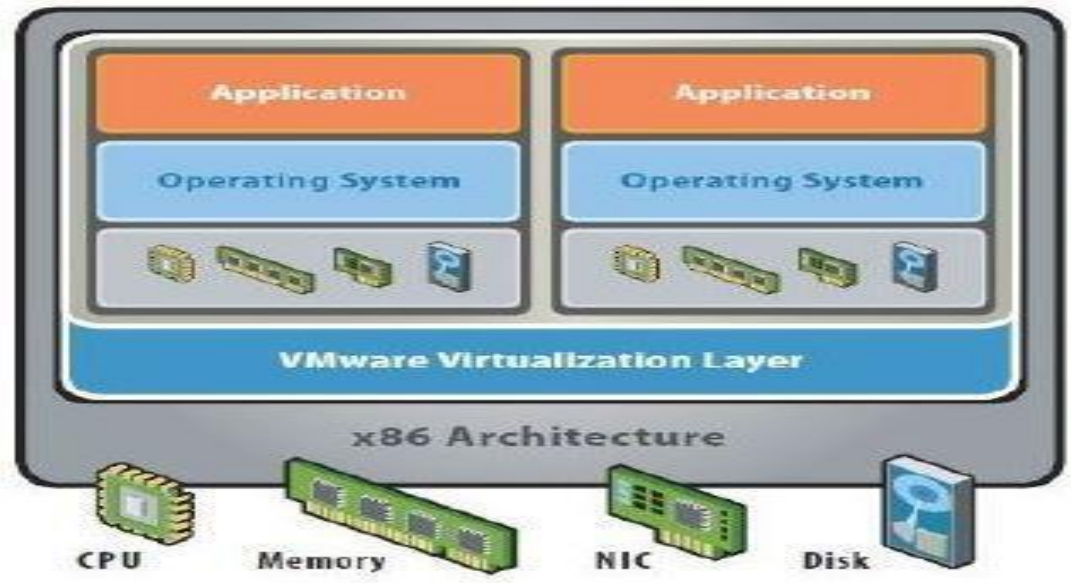



Figure 2. 3 Virtualization architecture [21].

In the case of cloud computing, virtualization is categorized into two:

**a. Full virtualization:** In this virtualization type, a complete installation of a certain machine is done on another machine and a virtual machine will have the whole software which present in a real server. It also helps a number of users to share a computer system [20].

**b. Para virtualization:** this virtualization type allow the underling hardware to run a number of operating systems on single machine. In para virtualization it is also possible to use system resources efficiently like memory and processor [20].

## 2.3 Cloud Computing Issues and Challenges

The number of cloud service users increasing day by day which in turn requires better performance of cloud computing and there are issues and challenges that needs to be addressed such as security, data loss, phishing, privacy and other threats, either at the enterprise level or individual level that use the pooled computing resources in cloud computing. So, there must have novel techniques to reduce the effect of the endless dangers in the cloud computing environment and the other challenge is performance**,** in cloud computing poor performance can be caused by nonexistence of resources like disk space, limited bandwidth, lower CPU

speed, memory, network connections etc. The data intensive applications are more challenging to provide proper resources and poor performance can result in end of service delivery, loss of customers and reduce revenues. In addition to security and performance, load balancing is another major challenges of cloud computing which is the main focus of this research.

## 2.4 Load Balancing in Cloud Computing

Load balancing is one of the major issues in cloud computing environment and it is a technique to distribute the workload evenly across all the nodes in the whole cloud to avoid a situation where some nodes are heavily loaded while others are idle or doing little work. The increase in web traffic and different application in the web world is increasing day by day where millions of data are created every second and load balancing is needed on CPU load, memory capacity and network. If there is a failure of any node or host system in the network, it will lead to isolation of web resource in the web world. Load balancing in such situation should be able to provide availability and scalability [22].

Load balancing helps to achieve a high user satisfaction and resource utilization ratio in cloud computing enviroment, hence improving the overall performance and resource utility of the system. Load balancing is the process of reassigning the total loads to the individual nodes of the collective system to make the best response time and also good utilization of the resources [23]. And some measurement parameters that can be used to evaluate the load balancing techniques, which allow us to check whether the given technique or algorithm of load balancing is good enough to balance the load or not [27].

a. **Scalability:** is used to determine the ability of load balancing algorithm to perform load distribution for a system with any finite number of nodes. For efficient load balancing algorithm this metric should be improved [28].

b. **Resource Utilization:** is used to determine the utilization of cloud computing resources and for efficient load balancing algorithm this parameter must be maximized or optimized [28].

c. **Performance** is used to check the efficiency and effictivness of the system. This has to be improved at a reasonable cost, e.g., reduce task response time while keeping acceptable delays [28].

d. **Response Time:** is used to determine the amount of time taken by a particular load balancing algorithm to respond to a certain user requests. And for efficient load balancing, this parameter should be minimized. For minimum response time, the resource utilization should be improved [28].

e. **Overhead Associated:** determines the amount of overhead or complex computation involved during implementing a load-balancing algorithm. It is composed of overhead due to movement of tasks, inter-processor and intercrosses communication. This should be minimized so that a load balancing technique can work efficiently [28].

f. **Throughput:** is measurement parameters to determine the number of task executed in the fixed interval of time. To improve the performance of the system, throughput should be high [28].

g. **Point of Failure**: designed the system in a way that the single point failure does not affect the provisioning of services. Like in centralized system, if one central node is fail, then the whole system would fail, so load balancing system must be designed in order to overcome this problem [28].

h. **Data Center Processing Time**: is measurement parameter used to determine the amount of time taken by the data center to excute a single task. It should be minimum with effective resource utilization [28].

There are two main categories of load balancing algorithm depending on state of the system.

a. **Static:** In this type of load balancing algorithm, the incoming load is assigned to different virtual machines based on the virtual machines processing capabilities and it divides the traffic generated by the user from different locations equivalently between the virtual machine. This load balancing algorithm is only suitable in a homogeneous environments because it does not check the current status of the virtual machines or nodes [29].

b. **Dynamic:** in dynamic load balancing algorithm the incoming request is assigned to the virtual machines by considering the current status or run time condition of the virtual

machines because this algorithm collects information and run times conditions of machines [29]. Dynamic load balancing algorithm is divided into distributed and non-distributed algorithm. In distributed dynamic algorithm, load vector of each node builds independently. In non-distributed dynamic managing and distributing the load is done by a single node.

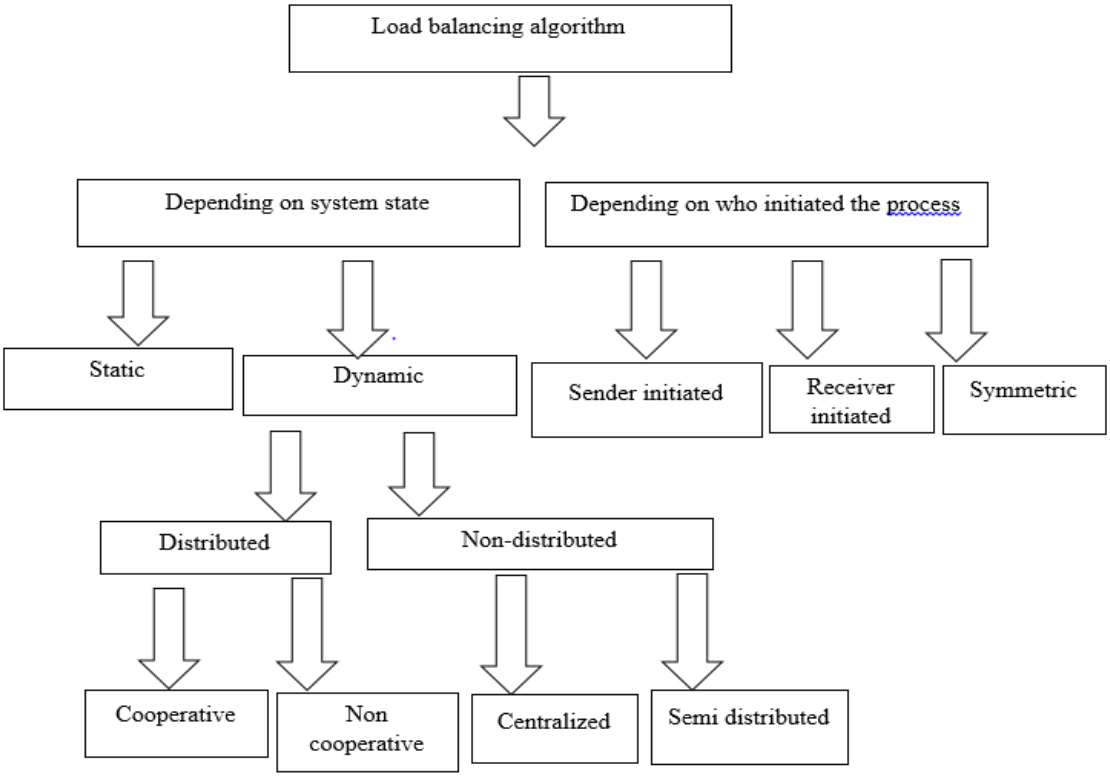



Figure 2. 4 Types of load balancing algorithm [40].

Under the architecture of cloud computing the load balancer is the most important parts of the environment. It distributes the incoming requesting using different techniques called load balancing algorithm. The load balancing algorithm starts its work after the request is arrived to data center and passes through data center controller. As figure 2.5 describes, there is a place for load balancing algorithm inside the cloud architecture so as to implement and insert the proposed algorithm for further benefit [9].

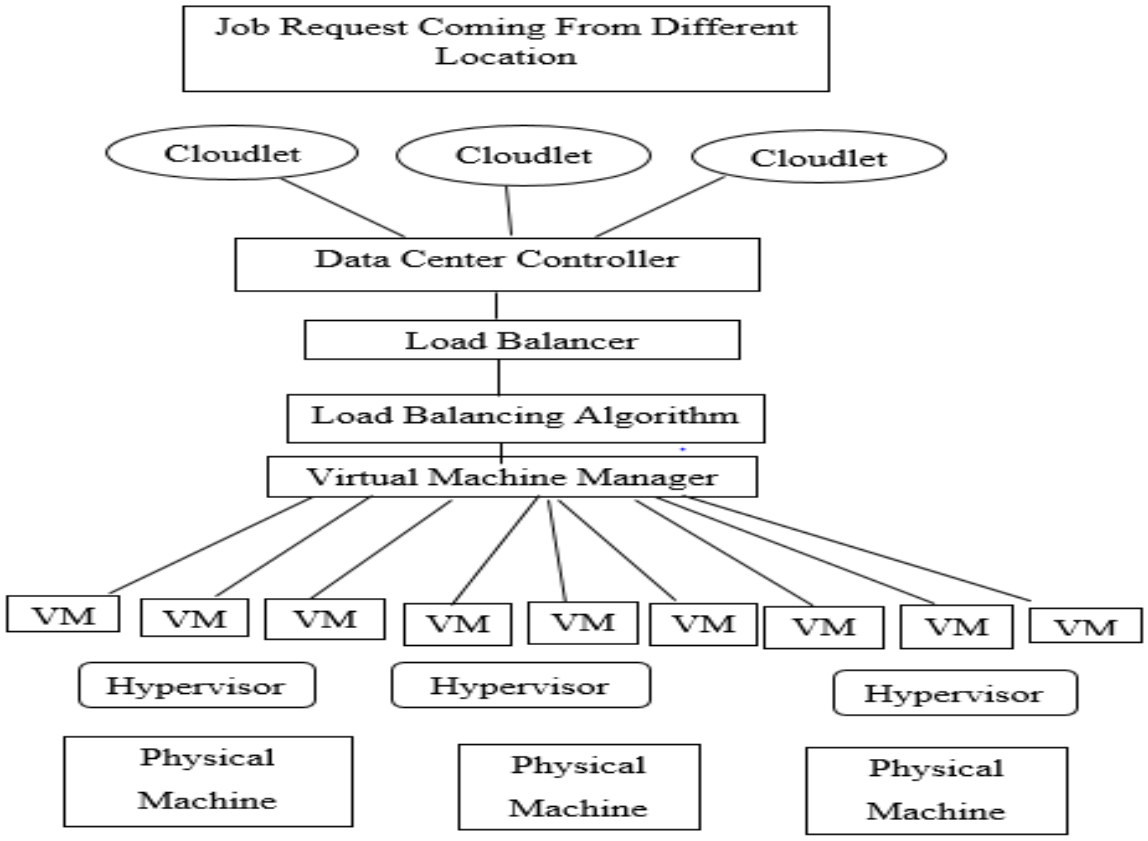



Figure 2. 5 Place of load balancer in cloud computing architecture [9].

## 2.4 Existing Load balancing Algorithm

Many researchers propose various load balancing algorithm to address load balancing issues of cloud computing and in this section we discuss about the existing well known load balancing algorithm with their advantage and drawbacks.

According to (Harish, Bahuguna) Round Rubin (RR) load balancing algorithm works in a round robin fashion to improve resource utilization [31]. When request arrives to the data center then the data center controller query round robin load balancer for appropriate virtual machine and round robin load balancer select virtual machine randomly for the first request but for the subsequent request, the virtual machine is selected in a circular order and once the virtual machine is assigned request it goes to the end of the virtual machine list.

The main advantage of this algorithm is work load distributions between processors are equal. And its main drawback is the job processing time for different processes are not the same and again if the VM is not free then incoming job should wait in the queue, which leads to

maximum response time because the overall execution time of the request will be the sum of waiting time and execution time.

According to (Isam, A, M) Random load balancing algorithm the incoming user request is assigned to the virtual machine randomly. when request arrives to the data center, the data center controller query random load balancer for appropriate virtual machine then load balancer peak the virtual machine randomly and send id of that virtual machine to the data center, the data center allocate the request to the virtual machine identified by that id. The advantage of this algorithm is that its complexity is quite low since it does not need any overhead or pre-processing in order to select and assign requests to the virtual machine. The main problem of this algorithm is it does not take into considerations the status of the VM which will either be under heavy or low load and this may result in the selection of a VM under heavy load so, in this algorithm resource utilization become not good and also the job requires a long waiting time before service is obtained [32].

The author in (Subhadra, Shaw, A.K) Throttled load balancing algorithm is used to assign the incoming cloudlet or user request to the available VM. In this load balancing algorithm, an index table having the id of VM along with its state either it is A V AILABLE or BUSY is maintained. When request arrives to data center then the data center controller queries the load balancer for allocation of the task. Then the load balancer scan index table from top until the first available VM is found or it will scan the index table fully. After that it will return id of that available VM to data center controller and assign the incoming user request to the VM identified by that id. After that the data center controller acknowledge load balancer for the allocation and the load balancer update the status of that VM to busy and when processing the task finished then the data center controller again acknowledge the load balancer about deallocation then the load balancer update status of the virtual machine to available. While processing the request, if no available VM is found then the load balancer return -1 to the data center controller and the task has to wait in queue. Advantage of throttled algorithm is it checks status of the virtual machine before assigning request and its main drawback is always scan the index table to find the available virtual machine from its first index each and every time and this maximize the response time [33].

According to the author (Manan D. Shah, Amit A. Kariyani, Dipak L. Agrawal) Active monitoring load balancing (AMLB) Algorithm assign user requests to the least loaded virtual machine. This algorithm maintains information about each VM and the current number of load each virtual machine holds. When a request to allocate a new VM arrives, data center controller queries the load balancer for allocation of the task and the load balancer scan index table from top to find least loaded VM among the list of virtual machine. Then it will return id of that least loaded VM to data center controller and the data center controller assign request to the virtual machine identified by that id after that data center controller acknowledge the load balancer about allocation then the load balancer increment current load of that virtual machine by one. When processing the request is finished then data center controller again acknowledge about the deallocation to the load balancer then load balancer decrement current load of that virtual machine by one. Advantage of AMLB is it maintains information about current load of each virtual machine. But its main drawback is when request arrives to data center it will scan the index table again and again from its first index to end of the index table to find least loaded VM as result of this associated overhead is high which in turn increase response time and in addition to this, AMLB it does not check whether the VM is recently used or not as result of this one VM is assigned in a continuous manner if it is least loaded and this will result in unfair distribution of load among the virtual machine [34].

Similarly (Jasmin James, Bhupendra Verma) improves Active monitoring VM load balancing algorithm by adding the concept weight, which is known as Weighted Active monitoring load balancing algorithm and this algorithm uses the concept of weights in active monitoring. The VM are assigned varying (different) amount of the available processing power of server/ physical host to the individual application services. To these VMs of different processing powers; the tasks/requests (application services) are assigned or allocated to the most powerful VM and then to the lowest and so on according to its weight and its availability. The drawback of this algorithm is its complexity is high which in turn increase the response time [35].

The author (Garg, Gupta and Rakesh) also improve Active monitoring VM load balancing algorithm, which is known as Enhanced Active Monitoring Load Balancing (EAMLB) algorithm and in this load balancing algorithm request is assigned to least loaded and not recently used virtual machine. In EAMLB, two hash maps are created. First is vmStateList that manage the total number of available and non-allocated VMs and second is current

Allocation Counts that holds the VMs with their allocation counts. When request arrives to data center then data center controller (DCC) receives a new request then it queries to the load balancer for new allocation. The load balancer first checks in vmStateList either any VM is available or not. If VM is available then allocate the job to it, otherwise assign request to the least loaded VM which is not recently loaded in current allocation count table. If the least loaded VM is equivalent to last used one then load balancer allocate the job to next least loaded VM. Advantage of EAMLB algorithm is it considers the virtual machine which is allocated recently along with the least loaded. This algorithm compare its experimental result with round robin and active monitoring. From this result its response time is improved. Enhanced Active Monitoring Load Balancing (EAMLB) algorithm gives the benefit that one VM will not be allocated in continuous manner if it is least loaded but its main problem is it starts scanning from the first index of the table again and again to find the least loaded VM and not last used VM as result of this the overhead computation becomes high and this in turn maximize response time [36].

Another author (S. G. Domanal, and G. R. M. Reddy) proposed modified throttled algorithm. It also maintains an index table containing a list of virtual machines and their states. The first VM is selected in the same way as in Throttled. When the next request arrives, the VM at index next to already assigned VM is chosen depending on the state of the VM and the usual steps are followed, unlikely of the Throttled algorithm, where the index table is parsed from the first index every time the Data Center Queries Load Balancer for allocation of VM. It gives better response time compare to the previous one. But in index table the state of some VM may change during the allocation of next request due to deallocation of some tasks. So it is not always beneficial to start searching from the next to already assigned VM [39].

The work which is proposed by (Supreeth, S, and Shobha Biradar) is weighted Round Robin algorithm which is the modified version of Round Robin in which a weight is assigned to each VM so that if one VM is capable of handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the Data Center Controller will assign two requests to the powerful VM for each request assigned to a weaker one. The major issue in this allocation is same as that of Round Robin that is it also does not consider the advanced load balancing requirements such as processing times for each individual requests.

Table 2. 1 Summary of related work

| No. | Algorithms | Description | Cons | Remarks compared with proposed algorithm |
|---|---|---|---|---|
| 1 | Round-robin [31] | It works in a circular order<br>Resource utilization is better and also its complexity is quite low | Response time is maximum | Response time is better than round robin because incoming job should not wait in the queue |
| 2 | Random algorithm [32] | Request is assigned to the VM randomly Its complexity is quite low | It does not check current load of VM | The algorithm check current load of each VM before assigning requests |
| 3 | Active monitoring [34] | Maintain information about current load of each VM and request is assign to the least loaded VM also it checks current load of VM | Resource utilization is not good and response time is also maximum | Response time is good because it does not require any preprocessing to select a VM also resource utilization is better since it does not assign one VM continuously |
| 4 | Enhanced active monitoring[36] | Maintain information about each VM and request is assigned to the least and not recently load VM and it avoids using of one least loaded VM continuously | High overhead Association, response time is also maximum | Low overhead association and this results minimum response time |

## 2.5 Cloud Simulators

The main aim of simulator is to test the implementation work in the absence of the required real environments [35]. The following are list of simulators used in the cloud computing environment.

a. **CloudSim:** is one of cloud simulator which is developed in University of Melbourne to perform seamless modelling, simulation and experimenting on designing Cloud

computing environment. It also enable us to model data centers, service brokers, scheduling and allocation policies of a large scaled Cloud platform because cloudSim is a self-contained platform. This simulator is built on top of GridSim framework [16]. One of the major drawbacks of CloudSim is lack Of GUI (Graphical user interface) [37].

b. **GridSim:** this framework is introduced in order to solve the impossibility of evaluating in a large scale distributed environment and which is developed by Buyya et al. This toolkit enable us to model and simulate heterogeneous Grid resources [26].

c. **iCanCloud:** which is developed by considering the drawbacks of CloudSim. It's very supple and the unique feature of this simulator is it allows user to customize the core hypervisor class. C++ programming language used in this simulator [37].

d. **CloudAnalyst:** this simulator is built by extending cloudSim features and it is very easy to use because it provides a graphical user interface. CloudAnalyst also differentiate simulation set up environment and it helps to only focus on the parameters used for simulation purposes instead of programming technicalities. Again the simulator allows to perform simulations again and again by giving different parameter values [30].

Table 2. 2 Summary of cloud simulators

| **Cloud Simulators** | **Description** |
|---|---|
| GridSim [26,27] | Used to simulate heterogeneous Grid resources |
| CloudSim[16] | It is used to model data centers, service brokers, scheduling and allocation policies of a large scaled Cloud platform because CloudSim is a self-contained platform . Its main limitation is lack of GUI. |
| CloudAnalyst [37] | By adding new extensions on cloudSim like Application users, Internet, Simulation defined by time period, Service Brokers, GUI based output and Ability to save simulations and results in the form of pdf. |

# Chapter 3: Methodology

## 3.1 Introduction

In order to conduct our research different technique and process was used and in this section we discussed about those techniques or methodology in detail.

## 3.2 Literature Review

It is our first step to understand what have been done and what have not been done in the area of the research work and there are many research works that have been done to solve load balancing issues of cloud computing. So in our study we reviewed literatures which are mainly related to load balancing issues of cloud computing and after understanding the research gap we proposed load balancing algorithm for cloud computing by taking advantage of random algorithm and enhanced active monitoring load balancing algorithms.

## 3.3 CloudAnalyst Simulator

Two choices are available to test the performance of proposed work, the first choice is to use a real test and the second one is to use simulation tools. In this study cloudAnalyst simulator was used because using real test limits the experiments to the scale of the infrastructure and makes the reproduction of results an extremely difficult undertaking and time consuming to measure performance in real cloud environment. In addition accessing the real infrastructure will incurs payment, as result of this using simulation tool is the best option.

CloudAnalyst simulator which is built on the top of cloudSim and it enable us to test and evaluate the performance of cloud services. Response time and data center processing time act as a performance evaluation parameter [30]. The following list of cloudanalyst features makes the simulator more usable than the other cloud simulator

a. **Simple to use:** CloudAnalyst is easy in setting up as it comes with the java package and we need to simply double-click on the icon. In this, simulation experiment execution is the main key feature of cloudAnalyst simulation tool [36, 37].
b. **GUI based output:** the simulator display the experiment results in the form of tables, graphs and charts. These GUI based output helps to understand and identify the important outlines of the parameters easily and also helps in their comparison [36, 37, 38].

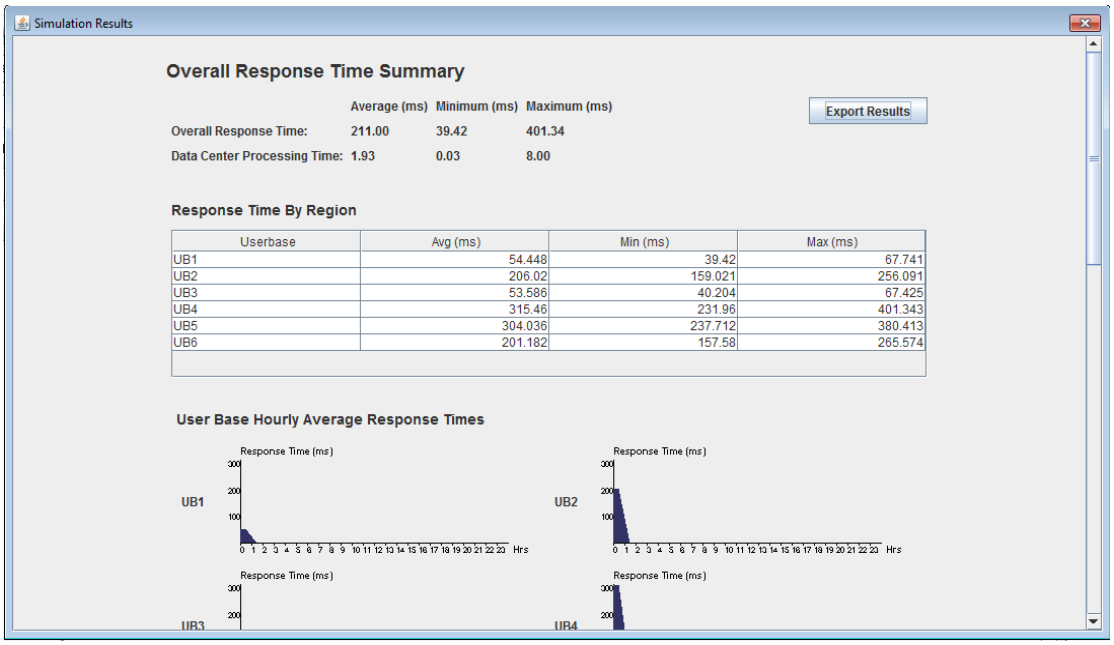



Figure 3. 1 Sample simulation output window [38].

c. **Ability to Repeat:** CloudAnalyst simulator enable us to repeat the experiment again and again. These is also a very significant requirement of any simulator because the simulator should produce the same results each and every time when the experiment is repeated many times with same data inputs otherwise the simulation just seems to be a random sequence of events and not the controlled experiment [36].

Cloudanalyst simulator has a very essential components which are used to model cloud computing environment and those components are listed as follows.

a. **Regions:** based on six different geographical locations or six continents of the world cloudanalyst simulator divides the simulation screen into six to consider the cloud users from those different region. This ecologically based categorization is very useful for maintaining the level of real-based simple and easy model for a very large duration simulation, which involves large parameters, are all performed in the CloudAnalyst and the results are obtained [37].

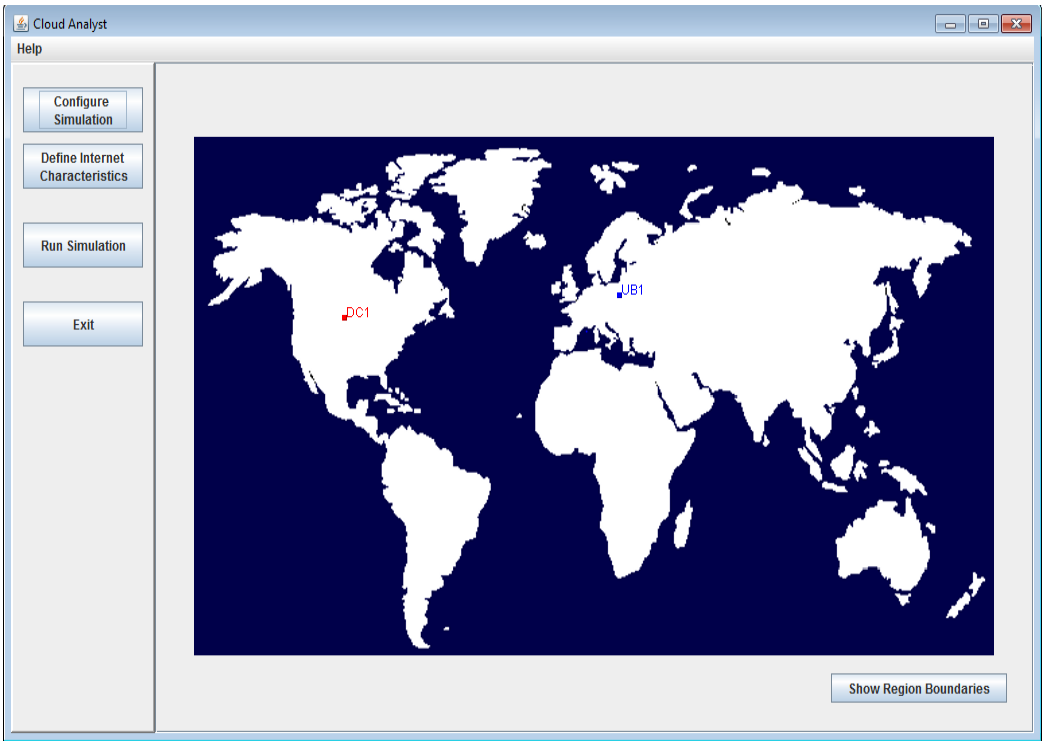



Figure 3. 2 Main screen with simulation panel [38].

**b. Internet Settings:** which is used to represent the actual real world Internet and it implements the whole features that are significant to a particular simulation. CloudAnalyst also manages to create the replicas of the Internet traffic, which seems to be routing around all over the sphere through hosting appropriate amount of data transmission delays and the transfer latency. The user can also configure the transmission latency and all the existing bandwidth among six areas (the whole world is divided into regions) [37].

**c. Service broker:** used to manage the traffic generated from the user base at different locations. In another word service broker is used to make a decision about servicing of the data center from each user base. And it three types of service brokers which implement different routing policies. The three service brokers are: closest data center which selects the shortest path from the user base to the data center, based on network latency and send the user request to closest data center by considering transmission latency, optimize response time to send user requests to the data centers which have best response time and dynamically reconfigured [37].

**d. User bases:** used to represent a collection of users from the six different continents of the world and which are considered as one individual unit in the cloudanalyst simulator. Its main responsibility is to generate traffic for simulation, with which the simulation is being considered as in the real time [37].

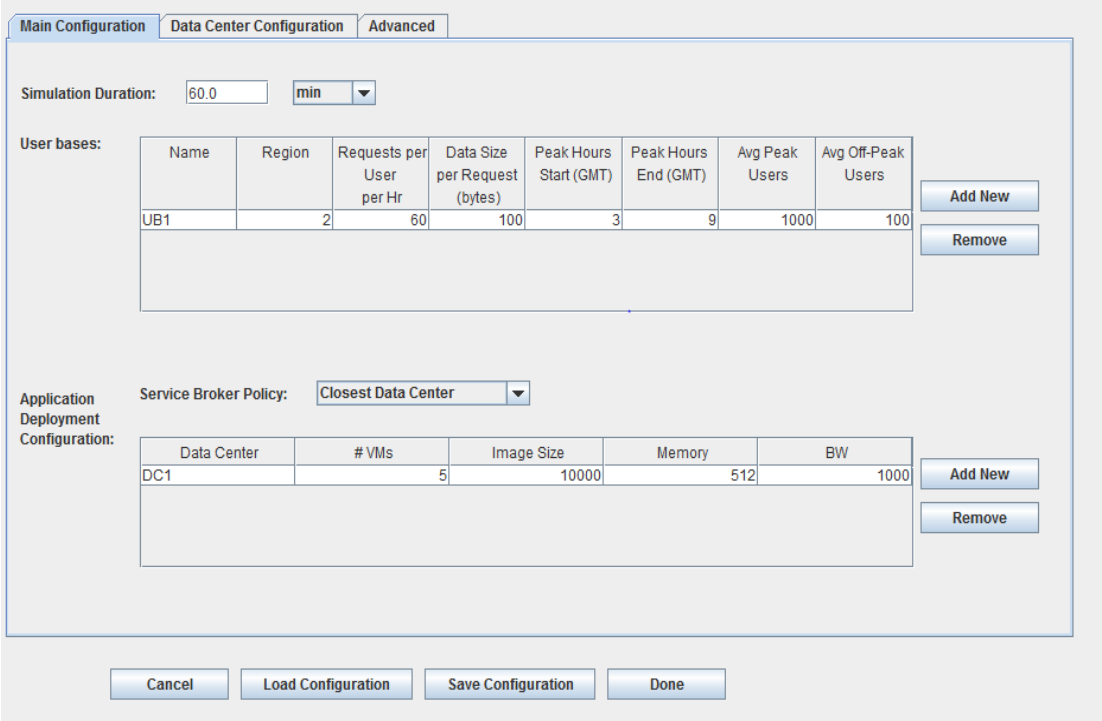



Figure 3. 3 User base configuration and application deployment configuration [38].

**e. Data center controller:** is help to manage the activities performed by the data center like creation of virtual machines, routes user requests that are received from respective user bases with the help of the Internet to the virtual machines. This can also be observed as the facade which is used by the cloudAnalyst for accessing and functioning the main part of cloudSim toolkit [37].

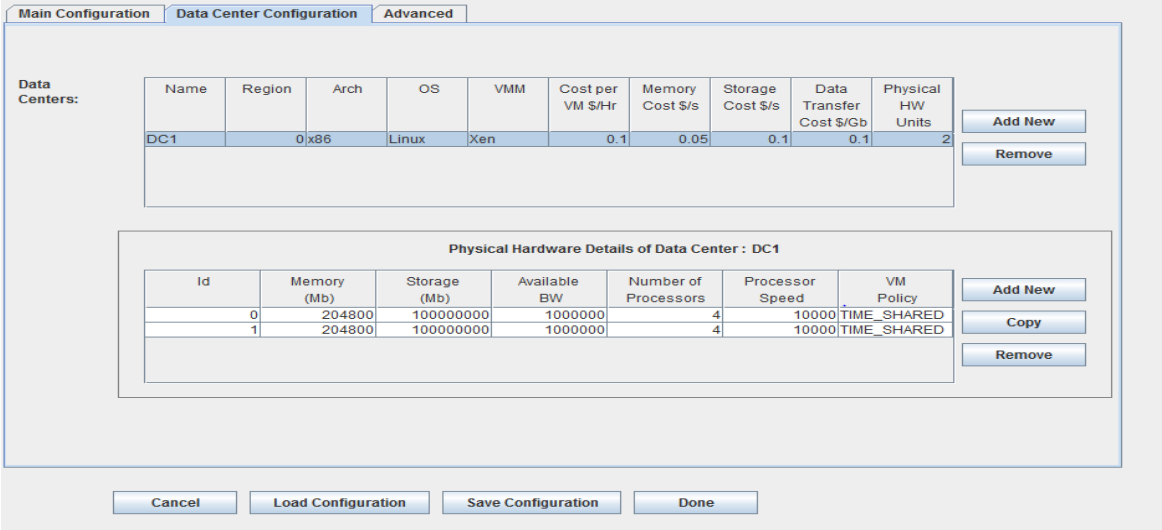



Figure 3. 4 Data center configuration in cloudanalyst [38].

**f. VM load balancer:** The data center uses the virtual machine load balancer in order to govern the particular virtual machines which are used to process the incoming user requests. Currently in cloudanalyst simulator three default VmLoadBalancers are available which implements three different load balancing procedures and their policies can be further selected by the modeler according to the requirement. The three different VM load balancers are active monitoring, round-robin and throttled load balancer [37].

**g. Application deployment configuration:** is a component which enable us to specify the number of virtual machines in the datacenter with their specification and these component includes the names of data centers created in the data center tab, number of VMs to be allocated to the application from the selected data center, image size a single VM image size in bytes, amount of memory available to a single VM, and amount of bandwidth available to a single VM [38].

**h. Advanced tap:** in this part of the cloudanalyst simulator there is a user grouping factor which is used to set the number of simultaneous users from a single user base and in the ideal scenario the value is on. The other one is request grouping which is used to set the number of simultaneous requests a single application server instance can support. In the ideal scenario this should be equal to 1. The last one in advanced tap is executable instruction length per request which is used to set the length of instruction.

Again load balancing policy which is the option that display available load balancing policy and it enable us to select the load balancing policy for the required activity.

In coloudAnalyst simulator routing of user request is done in nine steps [12]. Which are listed as follows:

1. Internet Cloudlet is generated from different user base, with the id of each application and name of the user base is also included for sending back the response of use request.
2. Send user request with zero delay to the Internet.
3. For appropriate data center selection, Internet first communicate with service broker. And then service broker can choose any the service broker policy by considering the incoming user request.
4. Information about the selected data center is send to the Internet by the service broker.
5. After receiving information about the data center, appropriate network delay is added to the request by the internet and sends back to to the data center controller.
6. Any one of load balancing algorithm for virtual machine assignment is used by the data center controller.
7. User request is assigned to the appropriate virtual machines to be processed by load balancer.
8. After processing the request, the Internet receive the response from the selected data center.
9. By using the originator field of the Cloudlet information, Internet adds appropriate network delay to the response and sends back to the user base.

### 3.4 Programming Language

After understanding cloudanalyst simulator the next move was developing proposed algorithm and for this java programming language is used because CloudAnalyst simulator architecture and its implementation is developed under java based environment [38]. In addition to this Java language has several features that is used for development and deployment of a software environment for cloud computing. Particular Java seem ideal for multi model communication environments. Java's platform-independent byte code can be performed firmly on a number

of platforms, making a programming language useful for portable cloud computing. Also, Java's ability on sequential codes has augmented considerably over the past years. Java offers a sophisticated interactive graphical user interface framework, as well as a model to invoke procedures on remote objects. So to develop the proposed load balancing algorithm first we integrate cloudanalyst simulator with eclipse java integrated development environment because it is very easy to import the available classes in the cloudanalyst simulator then proposed algorithm is developed.

### 3. 5 Testing and Evaluation

In order to test the efficiency and effectiveness of randomized enhanced active VM algorithm the simulation configuration was performed and with this simulation configuration it is repeated for round robin load balancing algorithm, active monitoring and enhanced active monitoring load balancing algorithms in addition to the proposed algorithm then compare simulation result of each algorithm with the proposed one.

# Chapter 4: Proposed Work

## 4.1 Overview

The current load balancing algorithm for cloud computing are not highly efficient and to improve the performance of cloud computing there is a need to address load balancing issue. The main objective of this research is to provide effective load balancing algorithm for cloud computing in order to overcome response time and resource utilization problem of load balancing. This chapter discuss about the proposed load balancing algorithm, including its implementation and evaluation metrics.

## 4.2 Proposed Algorithm Description

In this study a load balancing algorithm designed by taking advantage of the existing random and enhanced active monitoring load balancing algorithms. Random load balancing algorithm simply assign the incoming request randomly and it complexity is quite low to make decision for request assignment. But it does not check current number of request each virtual machines holds as result of this it does not select the best VM. On the other hand enhanced active monitoring load balancing algorithm select best VM which is least loaded and not recently used but the selection process is complex.

In proposed load balancing algorithm two hash maps are created. The first one is vmStateList in order to manage the total number of available and non-allocated VMs and the second one is current Allocation Counts to maintain information about the VMs with their allocation counts.

When request arrives to data center then the data center controller forward the request to the load balancer then the load balancer first check vmStateList either any VM is available or not. If VM is available then load balancer return id of available VM to data center controller and data center controller assign request to the VM identified by that id. But if there is no available VM in vmStateList hash map, the load balancer will select two VM randomly from current allocation hash map index and compare the current load of the two virtual machine then check the least loaded VM among the two virtual machine is recently loaded or not. If it is not recently loaded then select the least loaded one but if it is recently loaded, select the second VM. If both VM have equal number of request then the first VM is selected. After that the load balancer forward the appropriate VMs id to data center controller then data center

Controller assigns the incoming request to the selected VM and data center controller notify the load balancer about the allocation then load balancer update the selected VM allocation count by incrementing one on the current number of count. When the VM finishes the given request, it forwarded the response to data center controller then data center controller again notify about de-allocation to the load balancer and then load balancer update current count of that VM by subtracting one from the current count.

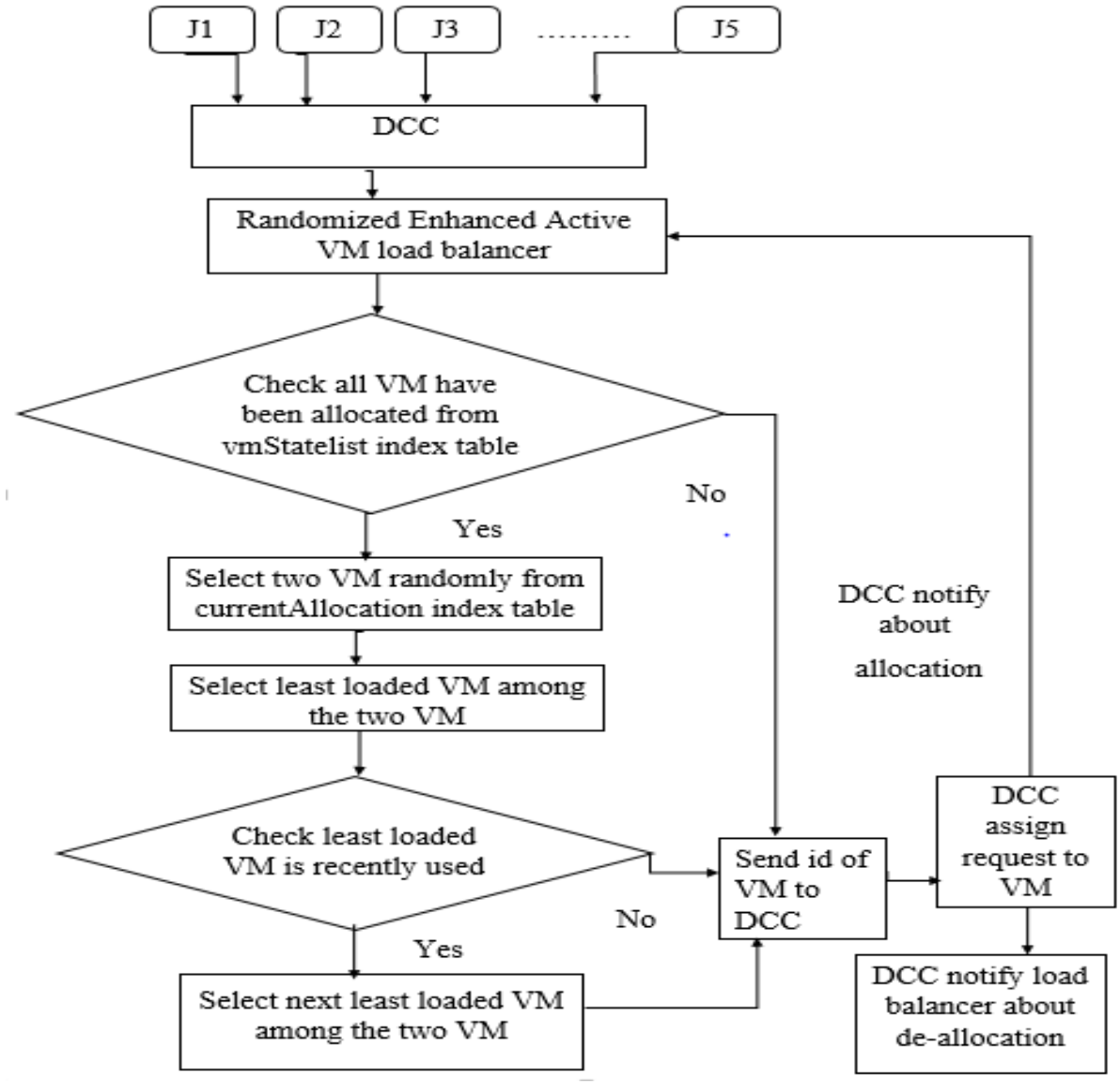



Figure 4. 1 Flow chart of developing randomized enhanced active VM algorithm

## 4.3 Implementation

The proposed load balancing algorithm has been implemented in CloudAnalyst simulator by programmatically extending the core framework provided in the CloudAnalyst. After implementing the proposed load balancing algorithm, configure the CloudAnalyst simulator for testing and evaluation. And in order to do this first we configure user base with their geographic distribution, and other properties like the number of users during peak and off-peak hour, peak hour time, number of request per hour and request size in byte in configuration window on main tab then we define the data centers we wish to use in the simulation with all the hardware and accounting aspects of the data centers in data Centers tab of the configuration screen. Once the data centers have been created, we allocate VMs in them for the simulated application using the main tab of the configurations screen. A data center defined above does not get included in the simulation unless it is allocated in this step after this, we review and adjust the advanced parameters in the advanced tab of the configuration screen and run the simulation after filling the required data and select the VM load balancer.

The algorithm is tested in homogenies environment and each virtual machine had the same bandwidth, processor speed, the same number of CPU and we tested the algorithm by considering the effect of varying the number of VM to handle the same user request and the effect of large number of user request from each user base. Finally we tested the efficiency of the algorithm in terms of the number of current allocation of each virtual machine. We also implement some of the existing load balancing algorithms (Round robin, Active monitoring and Enhanced active monitoring) and code of developed load balancing algorithm exists in appendix A.

## 4.4 Pseudo Code

The developed load balancing algorithm as given in Figure 4.2 is a load balancing algorithm used by the datacenter to distribute the received tasks efficiently over the virtual machine under a normal workload by finding the best VM among the group of VMs to assign the load in homogeneous cloud computing environment and it takes advantage of both random and enhanced active monitoring algorithms; the random algorithm which randomly select a VM to process the received tasks, does not need complex computation to make a decision but it does not select the best VM. On the other hand enhanced active monitoring algorithms

algorithm selects the best VM to handle the received task, but the selection process needs some complex computation to find the best VM. The research work in this study, considers the current load of the virtual machine. In this algorithm the load balancer first check vmStatelist index to find non-allocated virtual machine and if there is any non-allocated virtual machine then assign requests to that virtual machine but if there is no non- allocated virtual machine then the proposed algorithm load balancer check the currentallocation index table and randomly select two virtual machine after that, compare their current load and select the least load one then check the least load VM is recently used or not. If it is not recently used then send id of this VM to data center controller but if the least loaded virtual machine is recently used one then send id of the second virtual machine.

**Input:** Number of incoming requests (cloudlet) J1, J2, J3 ….Jn.
Available virtual machine VM1, VM2, VM3, …..VMn .
**Output:** All incoming requests J1, J2, J3 ….Jn are allocated to the available least loaded and not recently used virtual machine VM1, VM2, VM3,…VMn.

1. Create a hash map vmStatelist<vmid, VirtualMachineState> and currentAllocationCount<currentAllocationCount<vmid, curentActiveAllocationCount>
2. Data center controller receive new request
3. Randomized active VM load balancer check all VM have not been allocated from vmStatelist table

   Set VMID ← -1

   Set tempVMID ← -1
4. If currentAllocarioncount.size()<vmStateList.size() THEN
5. For each VM in vmStateList
6. If currentAllocationcounts does not contain VM THEN
7. Set vmId ← index value of checked VM
8. END if
9. END For
10. Else
11. Set currCount_1← 1

    Set currCount_2← 1
12. for i← 0 to i<currentAllocationCount.size THEN

    set Random r← new Random();

    set temp1← r.nextInt(currentAllocationCount.size())

    set temp2← r.nextInt(currentAllocationCount.size())

    set currCount_1← currentAllocationCounts.get(temp1)

    set currCount_2← currentAllocationCount.get(temp2)
13. if currCount_1<currCount_2 THEN
14. if temp1 is not tempVMID
15. VMID← index value of temp1
16. END IF
17. Else VMID←index value of temp2
18. END else
19. else
20. if temp2 is not tempVMID THEN
21. VMID← index value of temp2
22. END IF
23. Else VMID← index value of temp1
24. END else
25. END else
26. END if
27. END else
28. END For

Figure 4. 2 Pseudo code of proposed algorithm.

## 4.5 Evaluation Metrics

Different parameters are available to evaluate the performance of load balancing algorithms and in our research we used three measurement parameters.

1. Response time: it is the time taken by the load balancing algorithm to respond to a certain request from the client.
2. Resource utilization: a parameter used to identify resource usage of the load balancing algorithm.

## 4.5 Summary

In this chapter we presented the proposed load balancing algorithm which takes advantage of random algorithm and enhanced active monitoring algorithm. And it considers the current allocation count of the virtual machine and avoid scanning the index table again and again from its first index to find the appropriate virtual machine for the assignment of the incoming user request in order to improve the performance of the algorithm.

We implemented the proposed algorithm in a homogeneous cloud computing environment and we tested the algorithm by considering the effect of varying the number of VM to handle the same user request and the effect of large number of user request. Then compare the result with round robin, active monitoring and enhanced active monitoring load balancing algorithms. For evaluation we use response time, data processing time and resource utilization parameters.

# Chapter 5: Experiments, Results and Discussion

## 5.1 Introduction

In this chapter the discussion is about the experiment results which are conducted on this study in order to test efficiency of the proposed load balancing algorithm in terms of response time and resource utilization. For analysis we used the data from social network users because, the present generation has embraced the use of social networks in many roles both personal and commercial. In order to manage such a high traffic, cloud technology has been adapted.

Hypothetical applications like Facebook users, Twitter users, Internet users are considered for experimentation. Six different geographical locations (six different continents of the world) are considered [14]. Facebook is one of well-known social media and it had more than 2.19 billion active registered users on December 2017 which distributed across the six continents and the approximate distribution of registered Facebook user in each continents is listed as follows.

North America: 379 million of users;

South America: 266 million of users;

Europe: 340 million of users;

Asia: 935 million of users;

Africa: 177 million of users;

Oceania: 19 million of users;

In our experiment a single time zone is considered for all user locations and again we only consider one hundredth of the total Facebook users from each continents and it is assumed that only 1% of total users are online during peak hour and one tenth of peak hour users are considered as available during off peak hour for simplicity. And table 5.1 shows the data set which are used as simultaneous online user during peak hour and simultaneous online user during off-peak hour in each region for experimentation in this research.

Table 5. 1 Data sets used in the experiment

| User base | Region | Simultaneous on-line User during peak Hours | Simultaneous off-line User during peak Hours |
|---|---|---|---|
| UB1 | 0- North America | 37908 | 3791 |
| UB2 | 1- South America | 26658 | 2666 |
| UB3 | 2- Europe | 34089 | 3409 |
| UB4 | 3- Asia | 93542 | 9354 |
| UB5 | 4- Africa | 17701 | 1770 |
| UB6 | 5- Oceania | 1946 | 195 |

## 5.2 Experimentations

The experiment has two parts to test the performance of proposed algorithm in terms of response time and resource utilization parameters.

### 5.2.1 Experiment Part One

In the first experiment part, our focus was on evaluating performance proposed load balancing algorithm together with the existing well known load balancing algorithms by evaluating their response time and data processing time.

#### 5.2.1.1 Experiment 1

The first experiment is conducted by configuring one data center and 5 virtual machine in order to handle 2000 user request from each user base and the data size per request is 1000 bytes. The simulation duration is 30 minute in average. The simulated hosts have x86 architecture, virtual machine monitor Xen and Linux operating system. The simulation configuration is shown below.

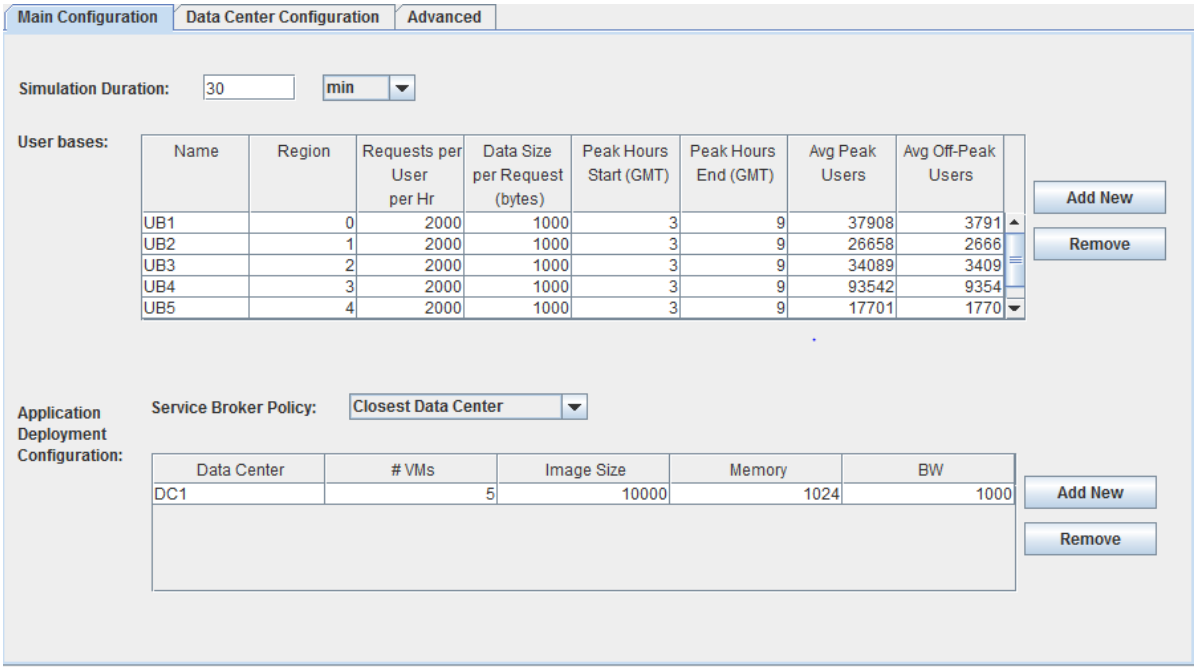



Figure 5. 1 User base configuration and application deployment configuration for experiment 1

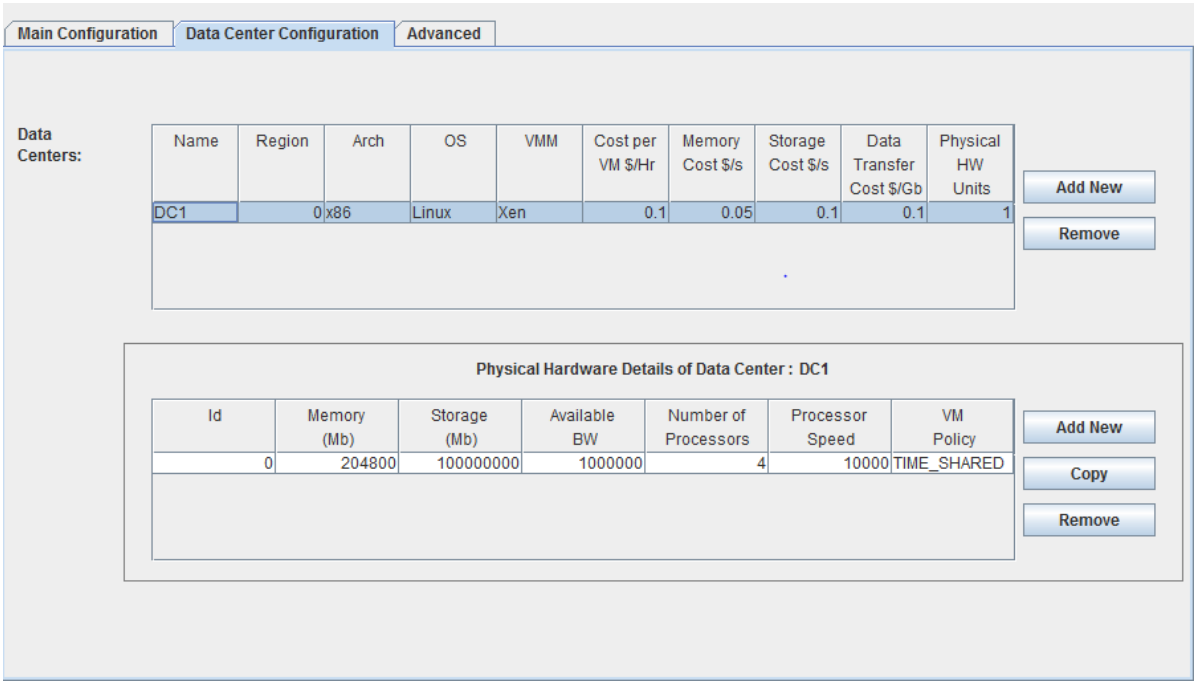



Figure 5. 2 Data center configuration for experiment 1

**Result**

In this experiment the simulation is repeated for round robin, active monitoring, enhanced active monitoring and developed randomized enhanced active VM. The following figures show the simulation result of overall response time summary for the four algorithms.

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 358.29 | 40.67 | 645.11 |
| Data Center Processing Time: | 3.50 | 0.03 | 17.86 |

Figure 5. 3 Round robin simulation result in experiment 1

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 358.29 | 40.67 | 645.11 |
| Data Center Processing Time: | 3.49 | 0.03 | 17.86 |

Figure 5. 4 Active monitoring simulation result in experiment 1

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 358.29 | 40.67 | 645.11 |
| Data Center Processing Time: | 3.49 | 0.03 | 17.86 |

Figure 5. 5 Enhanced active monitoring result in experiment 1

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 357.53 | 39.42 | 643.86 |
| Data Center Processing Time: | 2.69 | 0.03 | 17.10 |

Figure 5. 6 Randomized enhanced active VM algorithm result in experiment 1

**Average response time (milliseconds)**

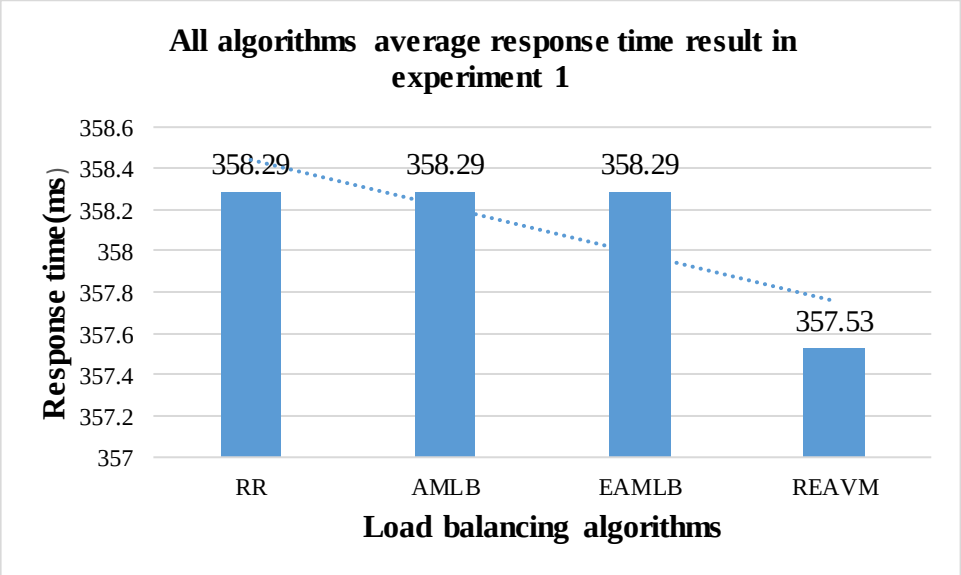



Figure 5. 7 Average response time in the case of 5 VM

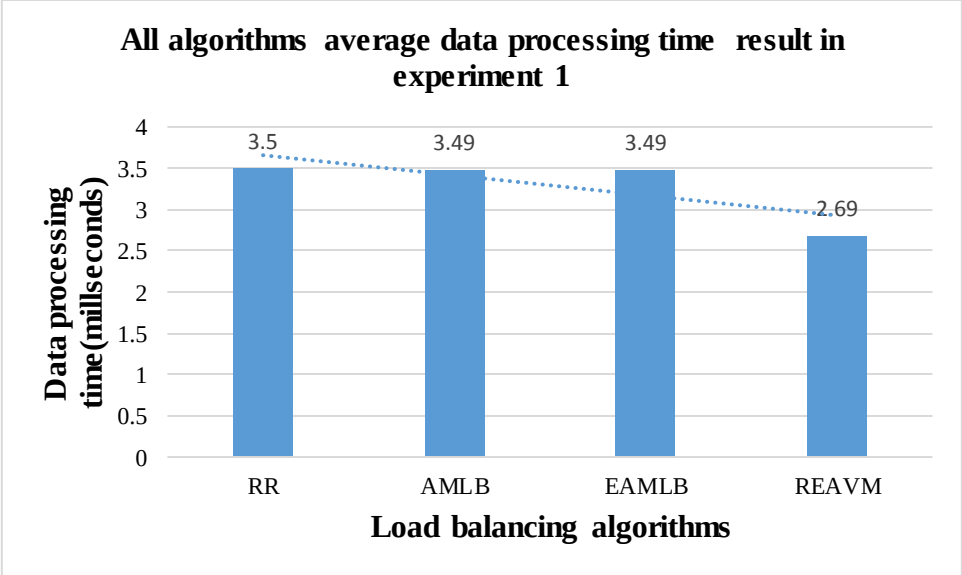



Figure 5. 8 Average data processing time in the case of 5 VM

**Discussions**

From this experiment result, Randomized enhanced active VM algorithm had better average response time 357.53 than Round robin, Active monitoring and Enhanced active monitoring algorithm because in the case of Round robin, it does not check the current load of the VM. Also in the case of active monitoring and enhanced active monitoring the overhead association is high. But in Randomized enhanced active VM algorithm, the distribution of load is fair because it checks the current load of the virtual machine. In addition to this the overhead computation is minimum to find least loaded and not recently used virtual machine.

We also found that Randomized enhanced active monitoring algorithm recorded better average data processing time 2.69 than Round Robin, Active monitoring and enhanced Active Monitoring load balancing algorithms because the complexity of randomized enhanced active VM algorithm is very simple to assign request or to take decision for VM allocation machine because it does not scan the index table again and again to find the least loaded and not recently used VM. So proposed load balancing algorithm is better in terms of response time and data processing time in cloud computing environment.

### 5.2.1.2 Experiment 2

The second experiment is conducted to test efficiency of randomized enhanced active VM load balancing algorithm together with Round robin, Active monitoring and Enhanced active

monitoring algorithm by configuring more number of data center and more number of virtual machine in order to handle the same user request from each user base as experiment one. In this experiment we configured two data center with 2 hardware unit at different region and 10 VM in each data center in order to handle 2000 request from each user base and the data size is 1000 and again the simulation duration is 40 minute in average. The simulated hosts have x86 architecture, virtual machine monitor Xen and Linux operating system. User base configuration, Application deployment configuration and data center configuration are shown in the following figures.

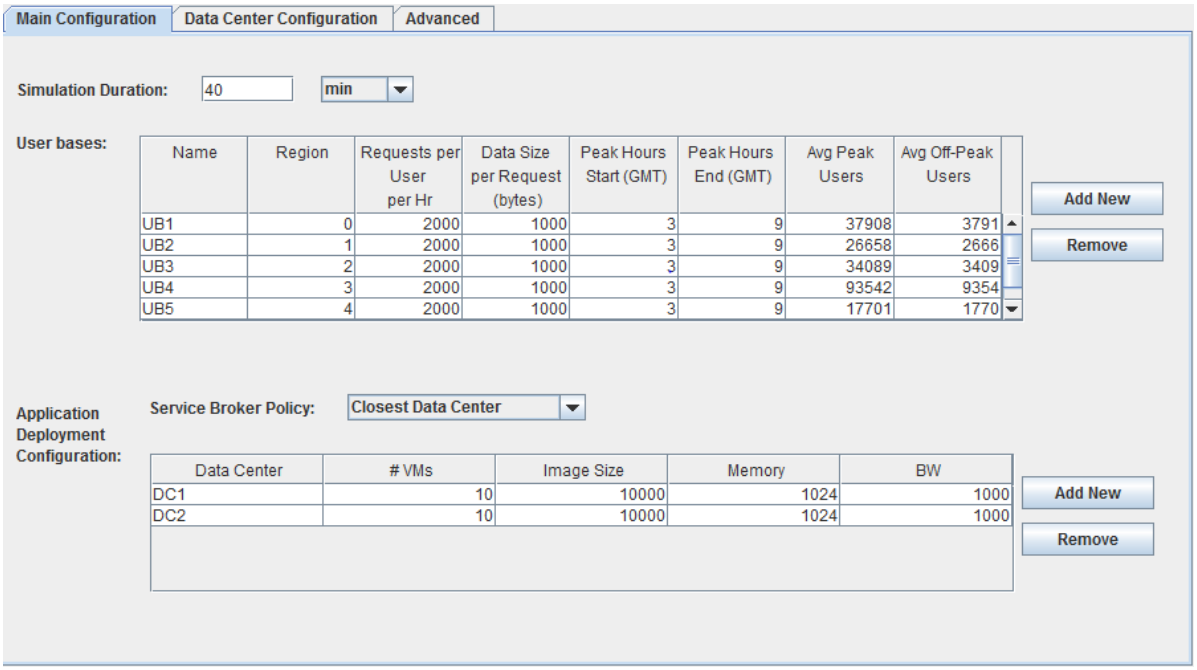



Figure 5. 9 User base and application deployment configuration for experiment 2

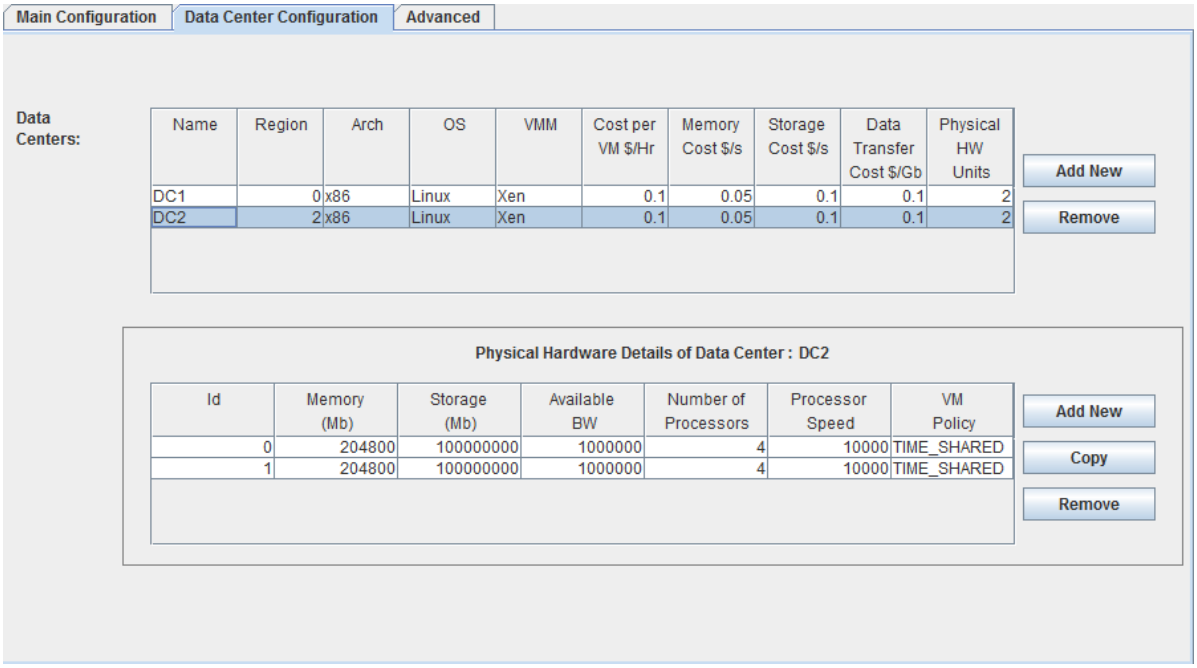



Figure 5. 10 Data center configuration for experiment 2

**Result**

In second experiment the simulation is repeated for round robin, active monitoring, enhanced active monitoring and randomized Enhanced Active VM. The simulation result is shown in the following figures for overall response time summary of the four algorithms.

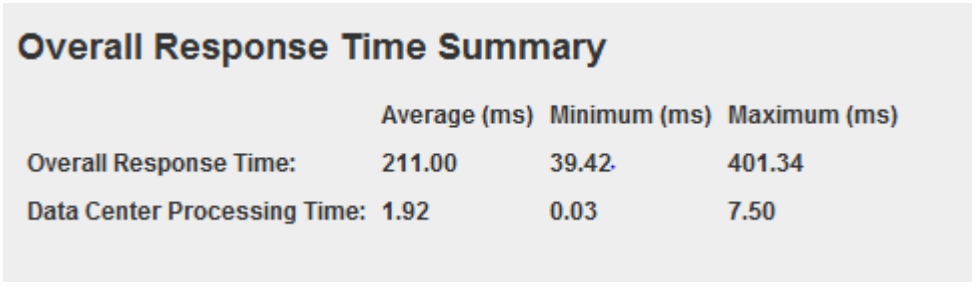



Figure 5. 11 Round robin algorithm simulation result in experiment 2

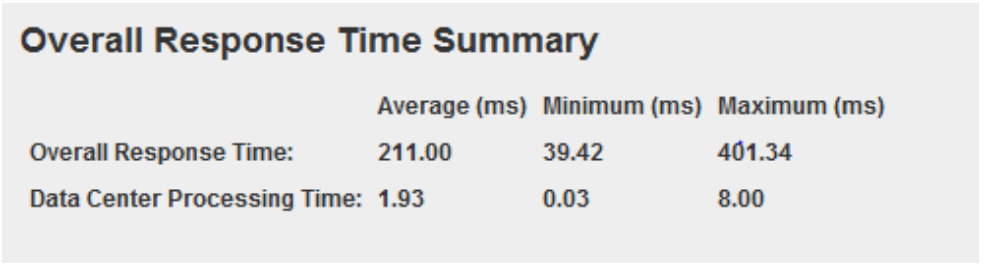



Figure 5. 12 Active monitoring algorithm simulation result in experiment 2

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 211.00 | 39.42 | 401.34 |
| Data Center Processing Time: | 1.93 | 0.03 | 8.00 |

Figure 5. 13 Enhanced active monitoring algorithm simulation result in experiment 2

**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 210.85 | 39.42 | 401.34 |
| Data Center Processing Time: | 1.77 | 0.03 | 8.35 |

Figure 5. 14 Randomize enhanced active VM algorithm simulation result in experiment 2

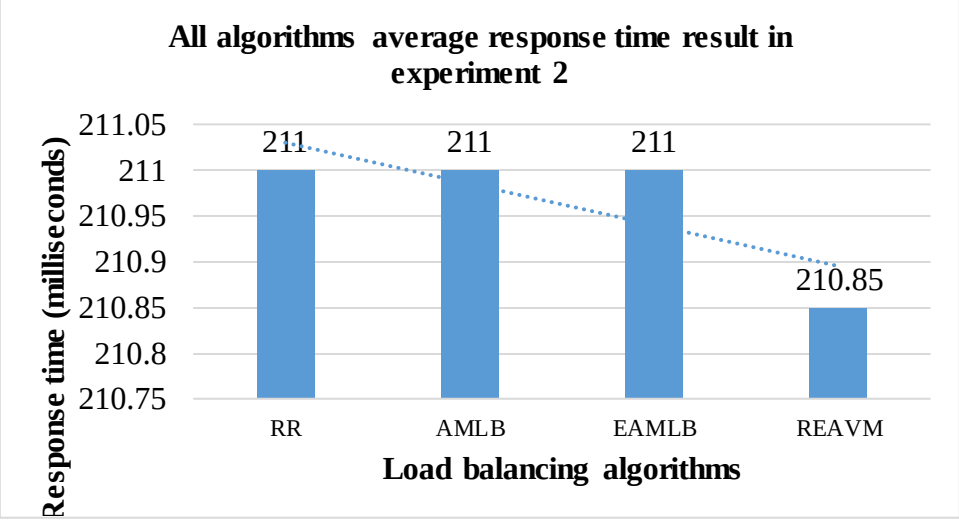



Figure 5. 15 Average response time in the case of 10 VM in 2 data center

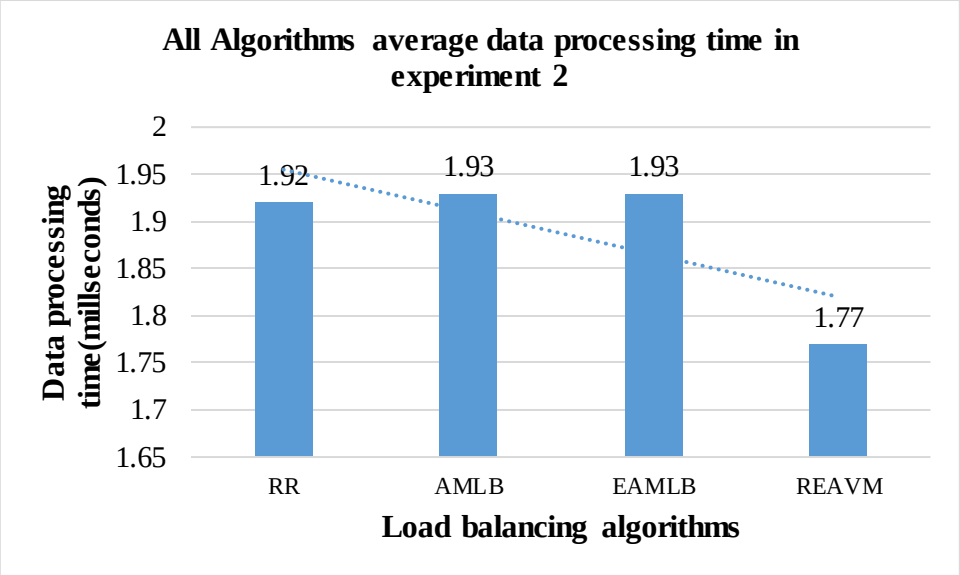



Figure 5. 16 Average data processing time in the case of 10 VM in 2 data center

**Discussions**

From the second experiment result Randomized Enhanced Active monitoring algorithm had better average response time 210.85 and average data processing time 1.77 than Round Robin, Active Monitoring and Enhanced Active Monitoring algorithm because it distribute the load fairly to all VM. In addition to this overhead computation is decreased even if the number of virtual machine is increased from 5 to 20 and the number of data center increased from 1 to 2 because randomized enhanced active monitoring does not require any preprocessing to assign request to the virtual machine and which in turn minimize the response time and data processing time. But the overhead computation in active monitoring and enhanced active monitoring is high to find the least loaded virtual machine because this two algorithm scan the index table again and again from its first index to find the least loaded virtual machine as result of this their average response time is not good as randomized enhanced active VM algorithm. In the case of round robin when there is no available virtual then incoming request waits in the queue and which in turn maximize its response time.

We also found that Round robin record better average data processing time than active monitoring and Enhanced active VM because the complexity of round robin is very small and it does not require any preprocessing to assign request to the virtual machine in a large number

of virtual machine. So in a large number of virtual machine proposed algorithm handles user request in a cloud computing environment.

#### 5.2.1.3 Experiment 3

The third experiment is conducted to test performance of proposed load balancing algorithm together with Round robin, Active monitoring and Enhanced active monitoring algorithm by configuring more number of data center and more number of virtual machine than experiment one. In this experiment we configured two data center with 2 hardware unit at different region and 10 VM in each data center in order to handle 10000 request from each user base and the data size is 1000 which is the same as experiment one and experiment two. Again the simulation duration is 40 minute in average. The simulated hosts have x86 architecture, virtual machine monitor Xen and Linux operating system.

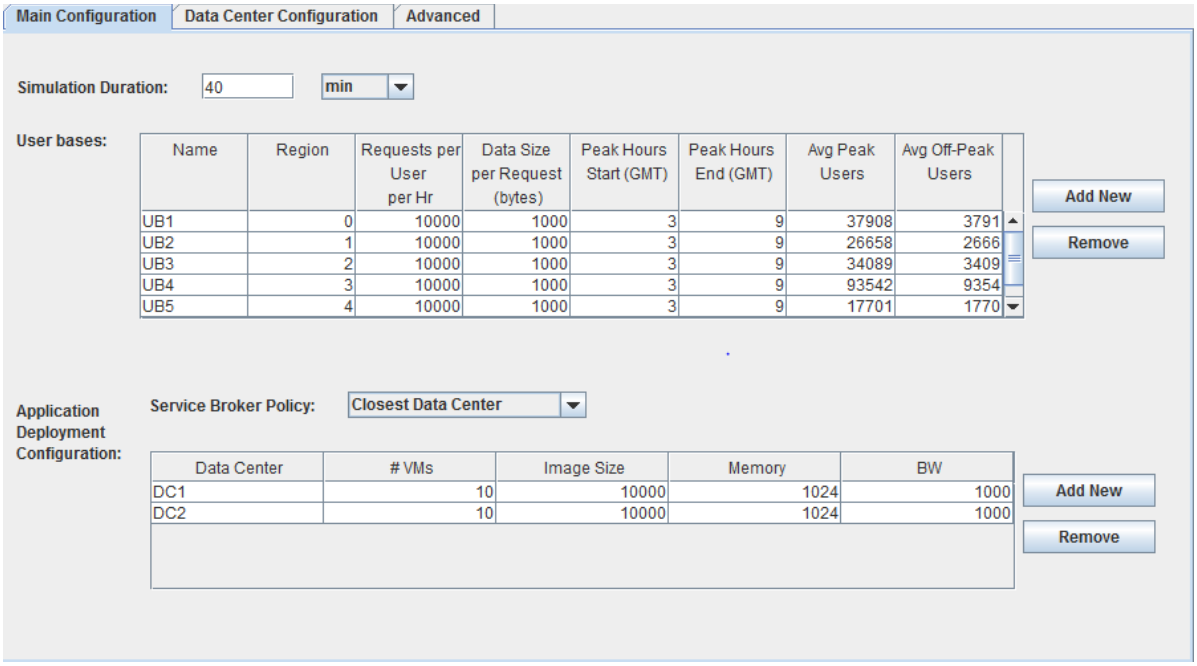
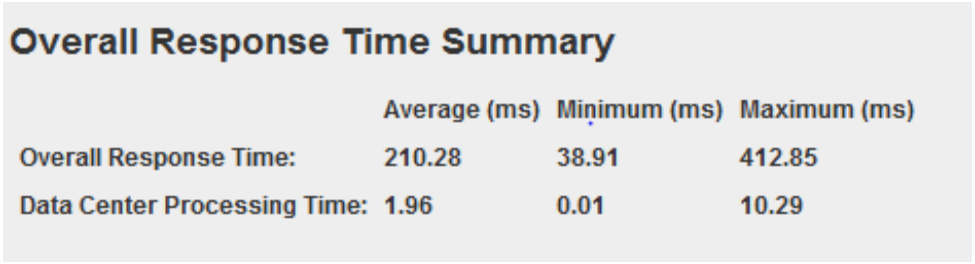



Figure 5. 17 User base and application deployment configuration for experiment 3

**Result**

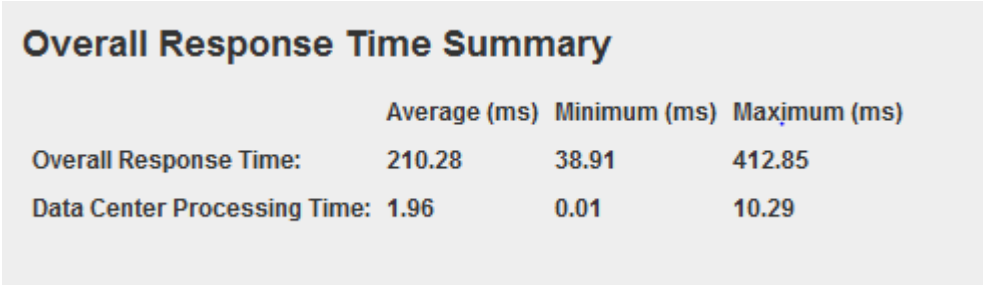
In third experiment, the simulation is repeated for round robin, active monitoring, enhanced active monitoring and randomized enhanced active VM. The simulation result is shown in the following figures for the overall response time summary of the four algorithms.



**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 210.28 | 38.91 | 412.85 |
| Data Center Processing Time: | 1.96 | 0.01 | 10.29 |

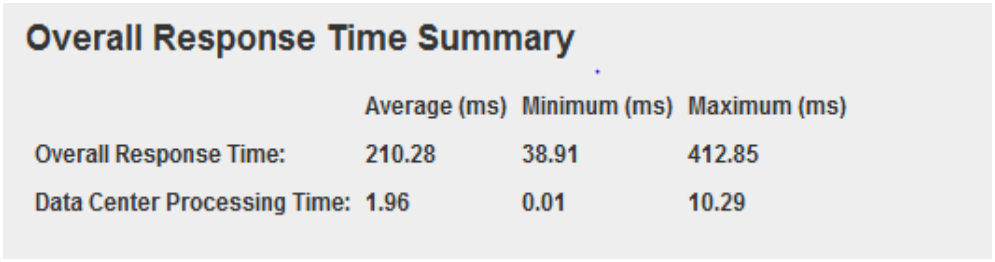
Figure 5. 18 Round robin algorithm simulation result in experiment 3



**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 210.28 | 38.91 | 412.85 |
| Data Center Processing Time: | 1.96 | 0.01 | 10.29 |

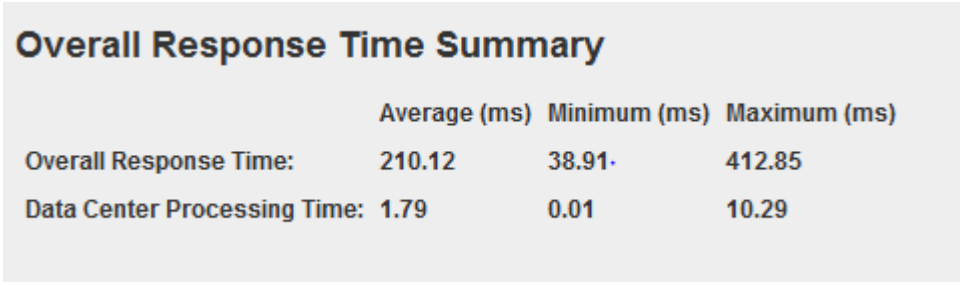
Figure 5. 19 Active algorithm simulation result in experiment 3



**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 210.28 | 38.91 | 412.85 |
| Data Center Processing Time: | 1.96 | 0.01 | 10.29 |

Figure 5. 20 Enhanced active monitoring algorithm simulation result in experiment 3



**Overall Response Time Summary**

| | Average (ms) | Minimum (ms) | Maximum (ms) |
|---|---|---|---|
| Overall Response Time: | 210.12 | 38.91 | 412.85 |
| Data Center Processing Time: | 1.79 | 0.01 | 10.29 |

Figure 5. 21 Randomized enhanced VM algorithm simulation result in experiment 3

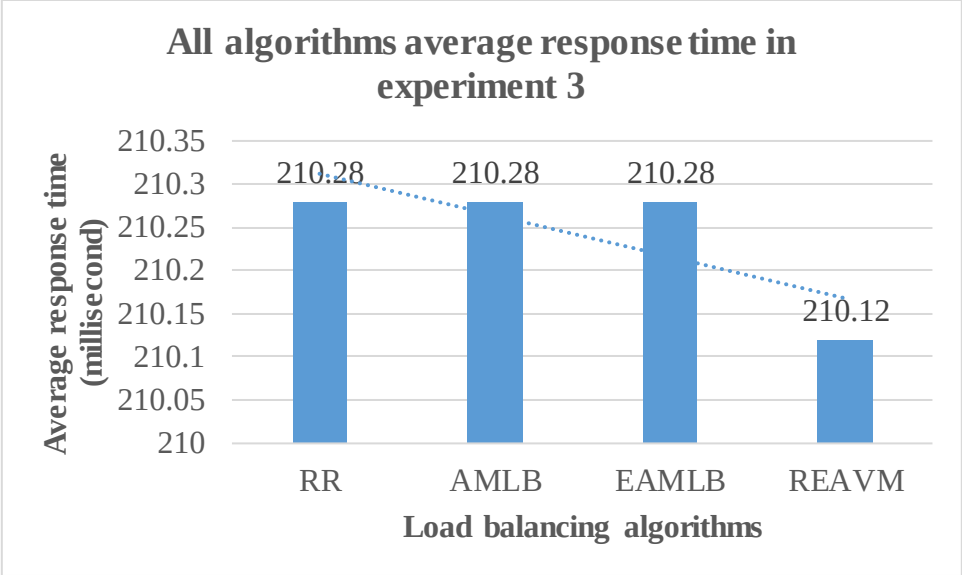



Figure 5. 22 Average response time in the case of 10000 user request

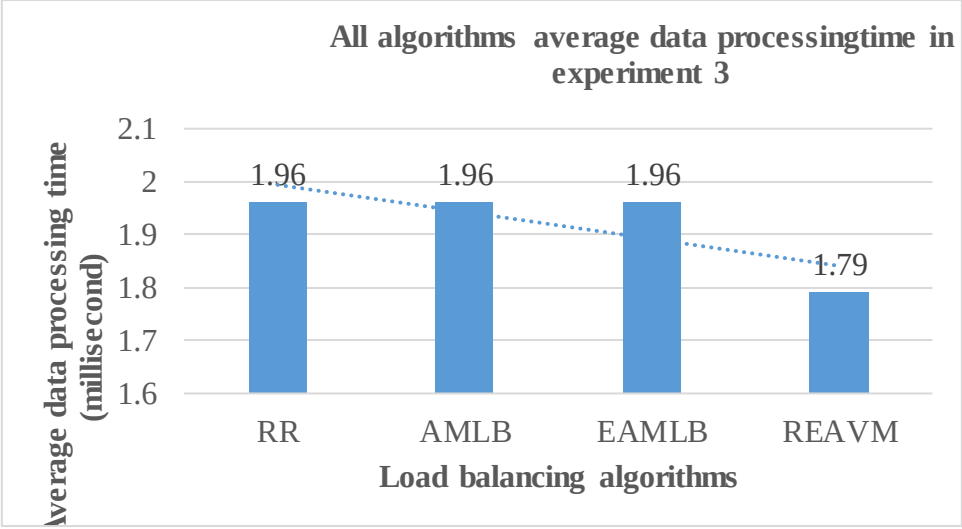



Figure 5. 23 Average data processing time in the case of 10000 user request

**Discussions**

From this experiment result randomized enhanced active VM algorithm had better average response time 210.12 than round robin, active monitoring and enhanced active monitoring algorithm. This was due to the efficiency of the algorithm. The algorithm became more efficient even when the number of request is large. Now days the social network process a big

data including big file and high resolution pictures so that the randomized enhanced active VM algorithm is more efficient for bigger request size than the other three algorithms.

In addition to average response time randomized enhanced active VM algorithm recorded better average data processing time 1.79 than round robin, active monitoring and enhanced active monitoring algorithms because the complexity of randomized enhanced active VM algorithm is very simple to assign request to the virtual machine because it does not scan the index table again and again to find the least loaded and not last used VM.

### 5.2.2 Experiment Part Two

In the second experiment part, our focus was on evaluating resource utilization of randomized enhanced active VM load balancing algorithm and the three well-known load balancing algorithms which are round robin, active monitoring and enhanced active monitoring algorithms.

#### 5.2.2.1 Experiment

The algorithms are tested for initially with 5 VM and later 10 VM. User base configuration, Application Deployment Configuration and Data center configuration are shown in the following tables.

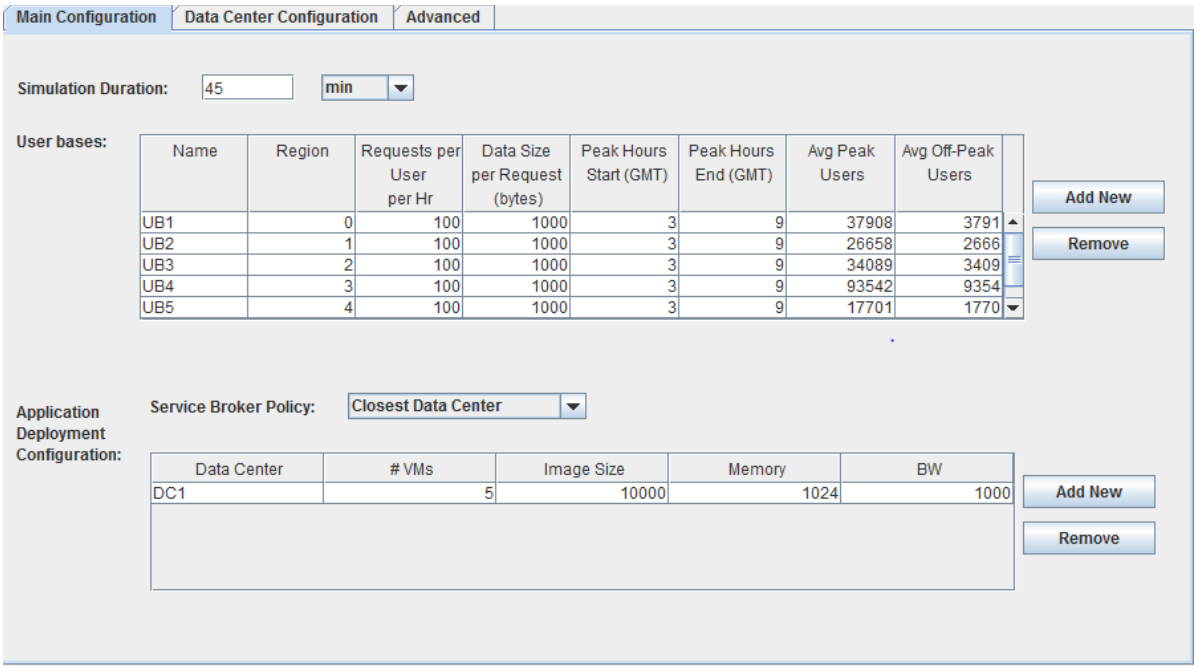



Figure 5. 24 User base and application deployment configuration for experiment 3

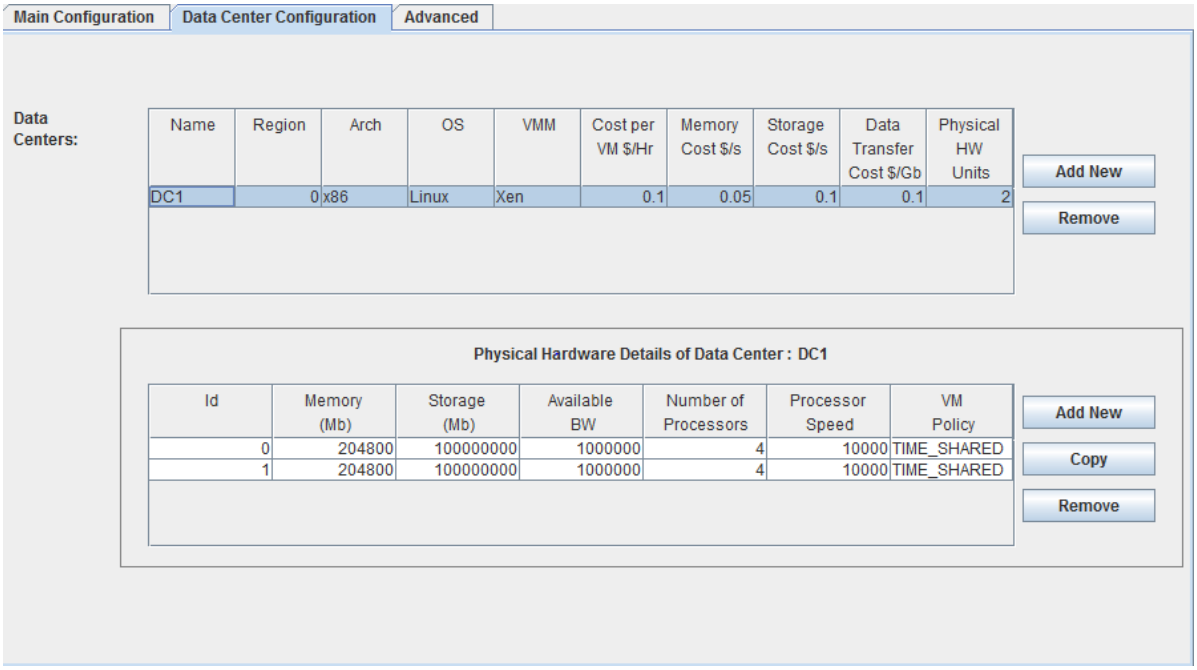



Figure 5. 25 Data center configuration

**Result**

The experiment with five virtual machine to test the efficiency of the algorithm in terms of load distribution among virtual machine is repeated for randomized enhanced active VM, round robin, active monitoring and enhanced active monitoring and the result is shown in table 5.12

Table 5. 2 VM allocation counts in experiment part two with 5 virtual machine

| **List of VM** | **Round robin** | **Active Monitoring** | **Enhanced active Monitoring** | **Randomized enhanced active monitoring** |
|---|---|---|---|---|
| VM0 | 6452 | 16822 | 16809 | 6837 |
| VM1 | 6452 | 3795 | 3799 | 6317 |
| VM2 | 6452 | 4288 | 4293 | 6503 |
| VM3 | 6451 | 3558 | 3569 | 6213 |
| VM4 | 6451 | 3795 | 3788 | 6388 |

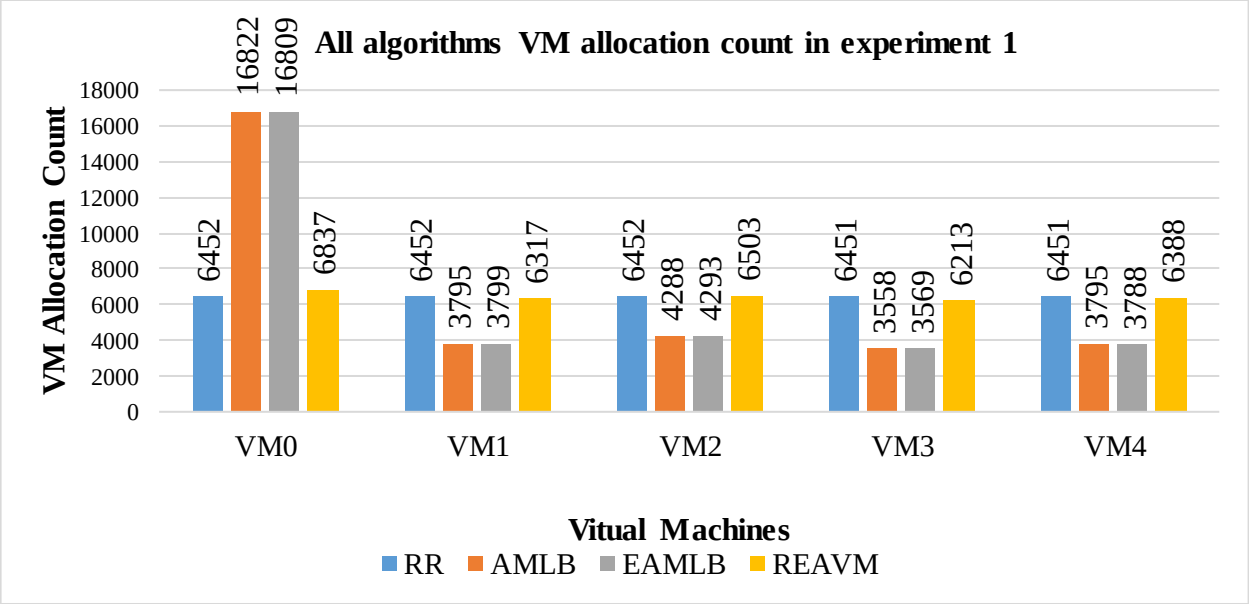



Figure 5. 26 VM allocation count in the case of 5 VM

Table 5. 3 VM allocation counts in experiment part two with 10 virtual machine

| List of VM | Round robin | Active Monitoring | Enhanced active Monitoring | Randomized enhanced active monitoring |
|---|---|---|---|---|
| VM0 | 3227 | 16461 | 16450 | 3378 |
| VM1 | 3227 | 1958 | 1961 | 3175 |
| VM2 | 3227 | 1922 | 1927 | 3422 |
| VM3 | 3226 | 1776 | 1777 | 3210 |
| VM4 | 3226 | 1829 | 1828 | 3331 |
| VM5 | 3226 | 1728 | 1727 | 3151 |
| VM6 | 3226 | 1745 | 1743 | 3311 |
| VM7 | 3226 | 1680 | 1691 | 3050 |
| VM8 | 3226 | 1638 | 1633 | 3222 |
| VM9 | 3226 | 1526 | 1526 | 3013 |

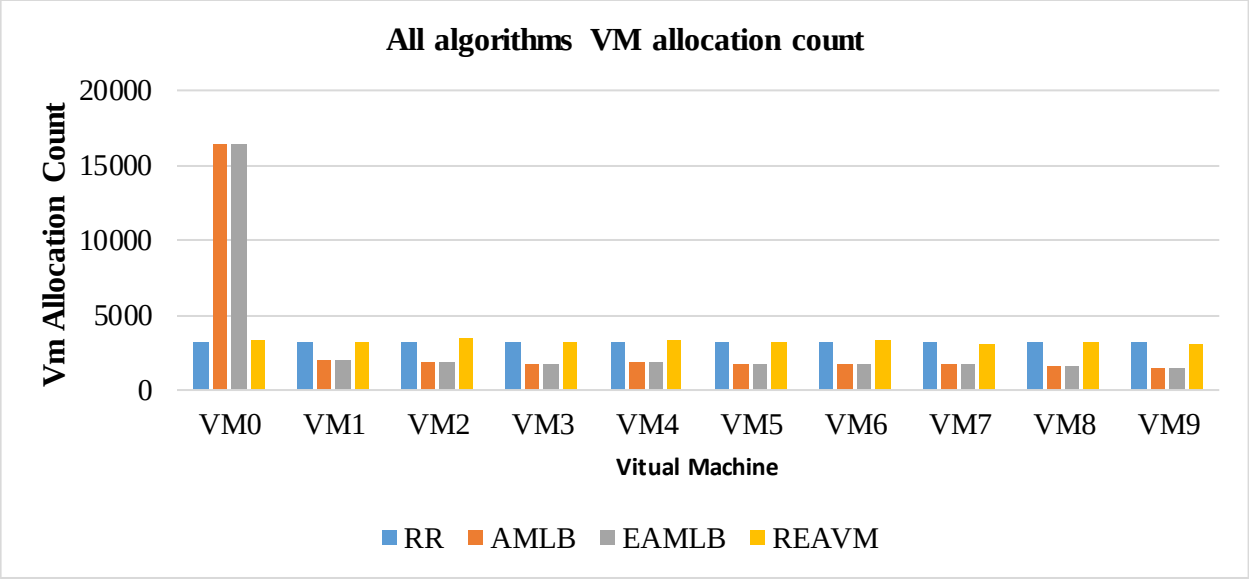



Figure 5. 27 VM Allocation Count in the case 0f 10 VM

**Discussion**

From the above two experiments randomized enhanced active VM algorithm and round robin algorithm distributes the load fairly to all virtual machine in both experiments but in active monitoring and enhanced active monitoring distribution of the load is not fair because as we have seen from the above table in the case of 5 VM the first virtual machines are heavily loaded 16822 in active monitoring and 16809 in enhanced active monitoring load balancing algorithm but some others are lightly load, as result of this resource utilization of the two algorithms are not good but utilization of the resources in randomized enhanced active VM algorithm is neither underutilized nor over utilized because it distributes the incoming requests to all VM's intelligent way and hence all the resources are used efficiently.

Also we found that Round robin distribute the incoming request fairly because it works in a circular order and this helps to avoid underutilization and overutilization of resource.

## 5.3 Summary

In the experimentation part of this document we tested and evaluated performance of proposed load balancing algorithm and repeated the experiments for Round robin,

Active monitoring, and Enhanced active monitoring and compare results we found in the experiment. For the experimentation we used cloudAnalyst simulator.

In this study the experimentation had two parts. In the first experiment part, we evaluate efficiency of randomized enhanced active VM load balancing algorithm in response time and data processing time perspectives. And in second part of experiment we evaluate the algorithm with load distribution viewpoint among the VM.

For the first experiment part, we consider three configuration by considering the number of data center, number of virtual machine and by considering large number of request from each user base. First we configured one data center with 5 virtual machine to handle 2000 user request from each user base. In second experiment we considered the configuration of two data center with 2 hardware unit and 10 virtual machines in every data center. In third experiment the data center configuration is the same as experiment two but the number of request is large that means we consider large number of request work load 10000 and in all the three experiment the size of data, time zone, peak hour user and off peak hour user are the same for all user locations. From all this simulation result, randomized enhanced active VM algorithm had better response time and data processing time than other algorithms because the proposed algorithm distributes the load uniformly and it is more efficient than others.

In the second part of the experiment we consider two configuration. The algorithm is tested for initially with 5 VM and later 10 VM. In both cases randomized enhanced active VM load balancing algorithm balances the load on all available VMs in an efficient way. Hence we can say that our algorithm will overcome the under or over utilization of resources usage problem. Also we found that round robin load balancing algorithm distribute the incoming load uniformly to the available virtual machine because, it works in a circular order.

# Chapter 6: Conclusion and Future Work

## 6.1 Conclusion

Many Researchers introduced different load balancing algorithm for cloud computing, like round robin, active monitoring, enhanced active monitoring, throttled etc. and those load balancing algorithms are not efficient for better performance of cloud computing as result of this there is need to have efficient and effective load balancing algorithm. In this study, a load balancing algorithm which handles the problem of response time and effective resource utilization for the incoming request has been introduced. The proposed algorithm takes advantage of both random and enhanced active VM algorithms. The random algorithm which select a VM uniformly and enhanced active VM algorithm check the current load of VM and avoid the situation in which one VM is assigned in a continuous manner and assign request to the least loaded and not last used VM.

For the experimentation CloudAnalyst simulator had been used and a number of experiments had been conducted by considering the number of virtual machine and number of data center. For evaluation, three metrics had been chosen: Response time and Number of request allocated to each VM or resource utilization.

The experimentation in this study had two parts. In the first experiment part we focused to evaluate response time and in second experiment part we evaluated resource utilization of randomized enhanced active VM algorithm and compare the results we found with round robin, active monitoring and enhanced active VM algorithms.

In the first experiment part the results showed that randomized enhanced active VM algorithm had better response time as compared to the other three algorithms because the proposed algorithm does not require any preprocessing to select appropriate virtual machine.

Also in the second experiment part randomized enhanced active VM algorithm distribute the load fairly and there is no underutilization and overutilization of resource

On this research we conclude that randomize enhanced active VM load balancing had better resource utilization and better response time than round robin, active monitoring and enhanced active monitoring

## 6.2. Future Work

Load balancing is considered as a major issue in cloud computing and for better performance of the cloud efficient and effective load balancing is required and the proposed load balancing algorithm is good in terms of response time and resource utilization and also the effectiveness of the algorithm is tested in a homogeneous environment that means all VM had the same memory, bandwidth, storage and CPU capacity. So for the future we are going to test the proposed algorithm in heterogeneous environment and it is also good to improve the efficiency of the proposed algorithm by adding additional features. Here are suggestions to be done for the future in order to improve the existing load balancing algorithm.

- Checking request type and request size before assigning to the virtual machine.
- Considering routing between client node and data centers.
- Considering computing power of virtual machine for request processing.

# References


[1] Shubham Sidana, Anurag Gupta, Neha Tiwari, Inall Singh Kushwaha," NBST Algorithm: A load balancing algorithm in cloud computing," International Conference on Computing, Communication and Automation (ICCCA2016) .

[2] Shikha Garg, D.V. Gupta2, Rakesh Kumar Dwivedi," Enhanced Active Monitoring Load Balancing Algorithm for Virtual Machines in Cloud Computing," Proceedings of the SMART -2016, IEEE Conference ID: 39669 5th International Conference on System Modeling & Advancement in Research Trends, 25th_27'h November, 2016 College of Computing Sciences & Information Technology, Teerthanker Mahaveer University, Moradabad, India(2016).

[3] Ankit Kumar, Mala Kalra," Load Balancing in Cloud Data Center Using Modified Active Monitoring Load Balancer," 2016 IEEE.

[4] Athokpam Bikramjit Singh1, Sathyendra Bhat J, Ragesh Raju, Rio D'Souza**,"** Survey on Various Load Balancing Techniques in Cloud Computing," Advances in Computing 2017

[5] Tushar Desai, Jignesh Prajapati,"A Survey Of Various Load Balancing Techniques And Challenges In Cloud Computing,"International Journal of Scientific & Technology research ,vol. 2, no.11, november 2013.

[6] YoussefFAHIM,Elhabib BEN LAHMAR,El houssine LABRUI,Ahmed EDDAOUI,Sara OUAHABI,'The load balancing improvement of a data center by a hybrid algorithm in cloud computing,"2014 IEEE

[7] Mayur S. Pilavare, Amish Desai," A Novel Approach Towards Improving Performance of Load Balancing Using Genetic Algorithm in Cloud Computing," IEEE Sponsored 2nd International Conference on Innovations in Information Embedded and Communication Systems ICIIECS'15 2015 IEEE.

[8] Sidra Aslam, Munam Ali Shah," Load Balancing Algorithms in Cloud Computing: A Survey of Modern Techniques," 2015 National Software Engineering Conference (NSEC 2015)

[9] Dharmesh Kashyap, Jaydeep Viradiya." A Survey Of Various Load Balancing Algorithms In Cloud Computing," International Journal of Scientific & Technology research,vol. 3, issue 11, november 2014

[10] Reena Pan war, Bhawna Mallick," Load Balancing in Cloud Computing Using Dynamic Load Management Algorithm," 20 15 IEEE.

[11] Manan D. Shah, Amit A. Kariyani, Dipak L. Agrawal," Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm,"IRACST - International Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555, Vol. 3, No.1, February 2013.

[12] Rakesh Kumar Mishra , Sreenu Naik Bhukya," Service Broker Algorithm for Cloud-Analyst," Rakesh Kumar Mishra et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3) , 2014 .

[13] Rajesh George Rajan1, V.Jeyakrishnan2," A Survey on Load Balancing in Cloud Computing Environments," International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 12, December 2013.

[14] Shridhar G.Damanal and G. Ram Mahana Reddy," Optimal Load Balancing in Cloud Computing By Efficient Utilization of Virtual Machines," 2014 IEEE.

[15] Peter Mell and Timothy Grance, 'The NIST Definition of Cloud Computing,"National Department of Standards and Technology, USA, September 2011.

[16] Sajjan R.S, Biradar Rekha Yashwantra," Load Balancing and its Algorithms in Cloud Computing: A Survey ," International Journal of Computer Sciences and Engineering,2017,Vol.1,pp.2347-2693.

[17] Dothang Truong, "How Cloud Computing Enhances Competitive Advantages: A Research Model for Small Businesses " Fayetteville State University,2010.

[18] Nikhil Rajeshirke, Rohan Sawant, Sumeet Sawant, Hasib Shaikh," Load Balancing In Cloud Computing," International Journal of Recent Trends in Engineering & Research (IJRTER),Vol.03,Issue 03, March 2017

[19] Amandeep, Vandana Yadav, Faz Mohammad," Different Strategies for Load Balancing in Cloud Computing Environment: a critical Study," International Journal of Scientific Research Engineering & Technology,2014,Vol.3, Issue 1.

[20] T.Swathi, K.Srikanth, S. Raghunath Reddy, "VIRTUALIZATION IN CLOUD COMPUTING ," International Journal of Computer Science and Mobile Computing,Vol.3,Issue 5, May 2014

[21] Preeti Thakur, Manish Mahajan," Virtualization in Cloud Computing," International Journal of Recent Trends in Engineering & Research (IJRTER),Vol.3,Issue 10,2016

[22] Qi Zhang, Lu Cheng, Raouf Boutaba," Cloud computing: state-of-the-art and research challenges ," The Brazilian Computer Society ,2010.

[23] Minakshi Berwal, Chander Kant," Load Balancing in Cloud Computing," International Journal of Computer Science and Computing,Vol.6,No.2,April-Sep 2015.

[24] Soumya Ray and Ajanta De Sarkar," execution analysis of load balancing algorithms in cloud computing environment," International Journal on Cloud Computing: Services and Architecture (IJCCSA), 2012,Vol.2, Issue 5

[25] Rahul Rathore, Bhumika Gupta, Vaibhav Sharma, Kamal Kumar Gola," Implementation and Result Analysis of Priority Based Load Balancing In Cloud Computing," International Journal of Scientific & Engineering Research,2014,Vol.5, Issue 12.

[26] Amandeep Kaur Sidhu, Supriya Kinger," Analysis of Load Balancing Techniques in Cloud Computing," International Journal of Computers & Technology,2013,Vol.4, Issue 2.

[27] Hafiz Jabr Younis,"Effcient load balncing algorithm in cloud computing," MSC. Thesis, Islamic University,Gaza,2015.

[28] Dharmesh Kashyap, Jaydeep Viradiya," A Survey Of Various Load Balancing Algorithms In Cloud Computing," INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH,2014,Vol.3, Issue.11.

[29] Athokpam Bikramjit Singh, Sathyendra Bhat J, Ragesh Raju1, Rio DSouza," Survey on Various Load Balancing Techniques in Cloud Computing," Advances in Computing, 2017,Vol.2,pp.28-34.

[30] Mayanka Katyal, Atul Mishra," A Comparative Study of Load BalancingAlgorithms in Cloud Computing Environment," International Journal of Distributed and Cloud Computing,2013,Vol.1,No.2.

[31] Harish Chandra, Himanshu Bahuguna," A SURVEY OF LOAD BALANCING ALGORITHMS IN CLOUD COMPUTING ," International Journal of Computer Engineering and Applications,2017,Vol.11, Issue 12.

[32] Isam Azawi Mohialdeen," COMPARATIVE STUDY OF SCHEDULING AL-GORITHMS IN CLOUD COMPUTING ENVIRONMENT ," Journal of Computer Science,2013.

[33] Subhadra Bose Shaw,A.K. Singh," A Survey on Scheduling and Load Balancing Techniques in Cloud Computing Environment," 5th International Conference on Computer and Communication Technology,2014.

[34] Manan D. Shah, Amit A. Kariyani, Dipak L. Agrawal," Allocation Of Virtual Machines In Cloud Computing Using Load Balancing Algorithm," International Journal of Computer Science and Information Technology & Security,2013,Vol.3,No.1.

[35] Jasmin James, Bhupendra Verma," EFFICIENT VM LOAD BALANCING ALGORITHM FOR A CLOUD COMPUTING ENVIRONMENT," International Journal on Computer Science and Engineering,2012,Vol.4,No.09.

[36] Shikha Garg,D.V. Gupta, Rakesh Kumar Dwivedi," Enhanced Active Monitoring Load Balancing Algorithm for Virtual Machines in Cloud Computing," 5th International Conference on System Modeling & Advancement in Research Trends, 2016.

[37] Arif Ahmed, Abadhan Saumya Sabyasachi," Cloud Computing Simulators: A Detailed Survey and Future Direction ," 2014 IEEE International Advance Computing Conference.

[38] Bhathiya Wickremasinghe," CloudAnalyst: ACloudSim-based Tool for Modelling and Analysis of Large Scale Cloud Computing Environments," UNIVERSITY OF MELBOURNE ,2009.

[39] S. G. Domanal, and G. R. M. Reddy, Load Balancing in Cloud Computing using Modified Throttled Algorithm, Proc. IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), 2013, 1-5.

[40] R. Ramya, Kariyani, L. Arockiam, "A State-of-Art Load Balancing Algorithms in Cloud Computing," International Journal of Computer Applications, 2014, Vol.95, No.19.

[41] Rajwinder Kaur, Pawan Luthra, "Load Balancing in Cloud Computing," Association of Computer Electronics and Electrical Engineers, 2014

[42] Supreeth, S, and Shobha Biradar. 2013. "Scheduling Virtual Machines for Load Balancing in Cloud Computing Platform." 2(6): 437–41.

# APPENDICES

## I. Proposed Algorithm

```
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.Random;
import java.util.Set;

import cloudsim.VirtualMachine;
import cloudsim.VirtualMachineList;
import cloudsim.ext.Constants;
import cloudsim.ext.event.CloudSimEvent;
import cloudsim.ext.event.CloudSimEventListener;
import cloudsim.ext.event.CloudSimEvents;

public class RandomizedEnhancedActiveVMLoadbalancer  extends
VmLoadBalancer implements CloudSimEventListener {
     /** Holds the count current active allcoations on each VM
*/
     private Map<Integer, Integer> currentAllocationCounts;

     private Map<Integer, VirtualMachineState> vmStatesList;
     DatacenterController dcb;

     public RandomizedEnhancedActiveVMLoadbalancer
(DatacenterController dcb){
          dcb.addCloudSimEventListener(this);
*              evenly distributed among the VMs.
      */
     @Override

public int getNextAvailableVm(){
          int vmId = -1;
          int tempVMId = -1;// Create variable to hold id of
last used VM
```

```
this.vmStatesList = dcb.getVmStatesList();
          this.currentAllocationCounts =
Collections.synchronizedMap(new HashMap<Integer, Integer>());
          dcb.addCloudSimEventListener(this);
}
/**
* @return The VM id of a VM so that the number of active tasks on
each VM is kept
//Find the VM with least number of allocations

          //If all available vms are not allocated, allocated the
new ones
          if (currentAllocationCounts.size() <
vmStatesList.size()){
//Step 1 : This command for selecting non allocated VM
for (int availableVmId : vmStatesList.keySet()){
                    if
(!currentAllocationCounts.containsKey(availableVmId)){
                         vmId = availableVmId;
                         break;
                    }
               }//end of step 1
} else {
               int currCount_1,currCount_2; //create variable to
store current load of randomly selected VM
               int temp1,temp2; // create variable to store id of
randomly selected VM
//Step 2: This command for selecting two VMs randomly (random)
for (int i=0;i<currentAllocationCounts.size();)
{
Random r=new Random();
temp1=r.nextInt(currentAllocationCounts.size());
temp2=r.nextInt(currentAllocationCounts.size());
currCount_1 = currentAllocationCounts.get(temp1);
currCount_2 = currentAllocationCounts.get(temp2);
// compare  current load of the VM selected randomly
if(currCount_1<=currCount_2)
{
// check id of value of temp1 is different from last used VM
if(temp1!=tempVMId)
{// create assign value of temp1 to vmId
vmId=temp1;
break;
}
```

```
else
{                       // assign value of temp2 to vmId
vmId=temp2;
break;
}

}
else
{
// check id of value of temp2 is different from last used VM

if(temp2!=tempVMId)
{
vmId=temp2;
break;
}
else
{
vmId=temp1;
break;
}
}
}
} // end of step 2
tempVMId =  vmId; // update value of tepVMId
allocatedVm(vmId); // pass value of vmId

return vmId;

     }

     public void cloudSimEventFired(CloudSimEvent e) {
          if (e.getId() ==
CloudSimEvents.EVENT_CLOUDLET_ALLOCATED_TO_VM){
               int vmId = (Integer)
e.getParameter(Constants.PARAM_VM_ID);

               Integer currCount =
currentAllocationCounts.remove(vmId);
               if (currCount == null){
                    currCount = 1;
               } else {
                    currCount++;
               }
               //Update the Value for allocated VM
               currentAllocationCounts.put(vmId, currCount);

          }
```

```
else if (e.getId() == CloudSimEvents.EVENT_VM_FINISHED_CLOUDLET){
//this event indicate that the VM deallocated and the current
allocate count will decrease by 1;
                int vmId = (Integer)
e.getParameter(Constants.PARAM_VM_ID);
                Integer currCount =
currentAllocationCounts.remove(vmId);
                if (currCount != null){
                     currCount--;
//Update the Value for allocated VM
                     currentAllocationCounts.put(vmId, currCount);
                }
           }
     }

}
```

# Plagiagiarism Report